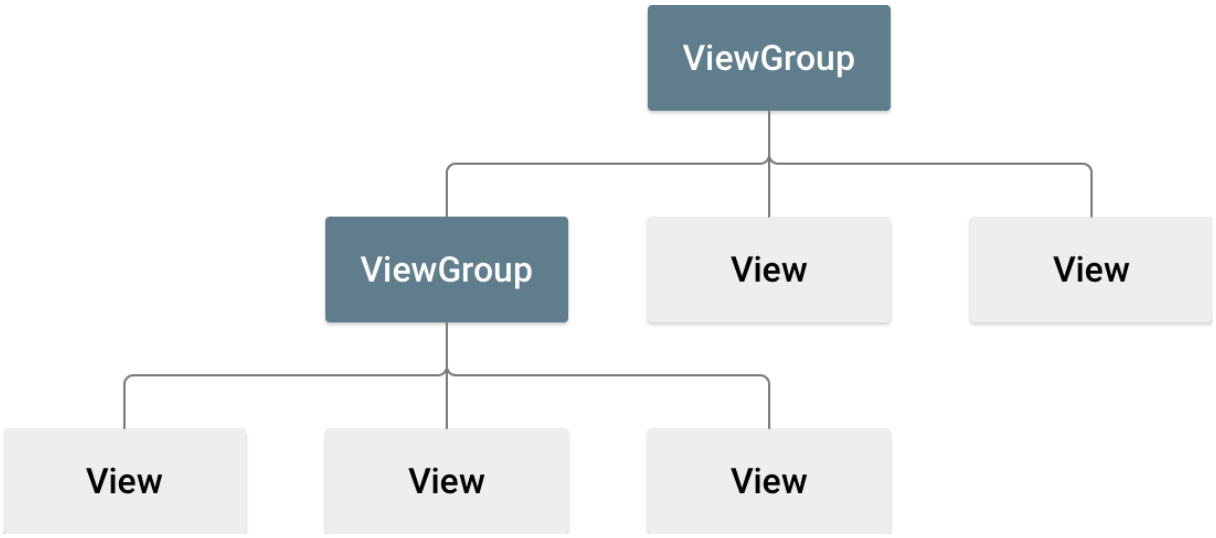
UI Design

In this document, we will be discussing the principles to use in order to create a sensible layout to your app, statically and dynamically.

What Is A Layout ?

A layout defines the structure of an interface as a tree of graphical components (View and ViewGroup objects), the root object being, in general, a container (ViewGroup object).



A ViewGroup is an invisible object (LinearLyout, RelativeLayout, …), whereas View objects are visible graphical components ususally called widgets (TextView, Button, …).

Such a tree can be declared either statically, dynamically, or with a mix of both.

Static Definition

The static definition of a layout is done through an XML file in which nested elements can be added to define the hierarchy of graphical objects.

Here follows an example of such a definition. Let us explain its content.

```
<?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" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

xmlns:android

defines a namespace named « android » which defines a list of tags that can be used by all the nested elements.

Usually, in any layout, nested elements are defined with specific attributes :

1. Attributes defining the position and dimensions of the element inside its parent element/container ; these are general attributes that should be given somehow for any element

2. Intrinsic attributes, such as a colour, a text, an id, … ; these attributes are totally dependent on the type of element being used

XML layout attributes named layout_*something* define layout parameters for the View that are appropriate for the ViewGroup in which it is contained.

android:layout_width
android:layout_height

These tags define the size of the element with respect to its container ; two possible values can be given ; **match_parent** to match the parent's size, or **wrap_content** to have the size of the element's content (text or whatever).

android:orientation

This tag is specific to the LinearLayout object and defines the way its going to place the nested objects (in a vertical way in this example)

android:id

This tag is common to all objects as it is inherited by all from the View class, and is optional ; it is needed whenever the programmer needs to access the elements, so it has to be identified in a unique way. The syntax to create an id is as follows :
@+id/name_example

To reference this object from inside the XML file, the syntax is :

@id/name_example

When processed, the framework generates the R class in which it creates a nested id class that contains this attribute (R.id.name_example). When the activity instantiates the objects, the programmer can then get a reference to one object by using the following function from the Activity class :

findViewById(R.id.name_example)

In the above example, to get a reference to the TextView object,

TextView myRef = (TextView) findViewById(R.id.text)

Layout Managers

Layout Managers handle the positionning and the dimensionning of nested elements. Some of the significant common attributes are given below :

Attribute	Description	Notes
layout_width	Specifies the width of the View or ViewGroup	
layout_height	Specifies the height of the View or ViewGroup	
layout_marginTop	Specifies extra space on the top side of the View or ViewGroup	
layout_marginBottom	Specifies extra space on the bottom side of the View or ViewGroup	
layout_marginLeft	Specifies extra space on the left side of the View or ViewGroup	
layout_marginRight	Specifies extra space on the right side of the View or ViewGroup	
layout_gravity	Specifies how child Views are positioned	Only in LinearLayout or TableLayout
layout_weight	Specifies the ratio of Views	Only in LinearLayout or TableLayout

LinearLayout

This layout manager places its nested elements serially, either vertically or horizontally.

```
<?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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
```

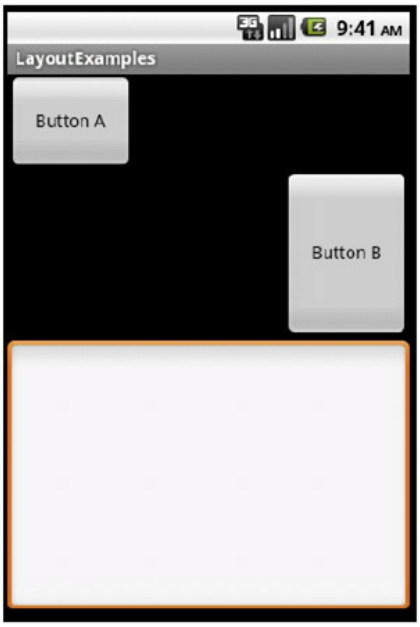
```
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

Paddings are used to add space between the content and its container.

Margins are used to add space between different nested elements.

When defining dimensions inside a container for the different elements, it's always better to give relative dimensions since the programmer does not know beforehand the size of the screen on which it is going to run. Therefore, child views can specify a weight value, and then any remaining space in the view group is assigned to children in the proportion of their declared weight. Default weight is zero.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:layout_width="match_parent"
      android:layout_height="match_parent"
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:stretchcolumns="*"
      >
      <Button android:layout_width="100px"
            android:layout_height="wrap_content"
            android:text="Button A"
            android:layout_weight="0.1"
            />
      <Button android:layout_width="100px"
            android:layout_height="wrap_content"
            android:text="Button B"
            android:layout_gravity="right"
            android:layout_weight="0.3"
            />
      <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:layout_weight="0.6"
            />
</LinearLayout>
```



Understand this last example and test it !

RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="reminder" />
    <EditText
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times"
        android:hint="e1"/>
    <EditText
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true"
        android:hint="e2"/>
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="done" />
</RelativeLayout>
```

Test this out. All The XML attributes are explained in the RelativeLayout.LayoutParams documentation.

TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:background="#646464">


    <TableRow>
        <TextView android:text="E-mail:"
            android:width="120dp"
            android:paddingLeft="20dp"/>
        <EditText android:id="@+id/email"
            android:width="200dp" />
```

```
    </TableRow>

    <TableRow>
        <TextView android:text="Password:" />
        <EditText android:id="@+id/password" />
    </TableRow>

    <TableRow>
        <TextView />
        <CheckBox android:id="@+id/rememberMe"
            android:text="Remember Me" />
    </TableRow>

    <TableRow>
        <Button android:id="@+id/signIn"
            android:text="Log In"
            android:layout_span="2" />
    </TableRow>
</TableLayout>
```

GridLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.edu.ck.myfirstapp.MainActivity"
    tools:showIn="@layout/activity_main">

    <!--
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
        -->
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Test"
        />

    <GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/gridLayout1"
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:columnCount="4"
        android:rowCount="4"
        android:layout_below="@id/button2">

        <TextView
            android:layout_width="150dp"
            android:layout_height="50dp"
            android:layout_column="0"
```

```
            android:layout_columnSpan="3"
            android:layout_row="0"
            android:background="#ff0000"
            android:gravity="center"
            android:text="one" />

        <TextView
            android:layout_width="125dp"
            android:layout_height="100dp"
            android:layout_column="1"
            android:layout_columnSpan="2"
            android:layout_row="1"
            android:layout_rowSpan="2"
            android:background="#ff7700"
            android:gravity="center"
            android:text="two" />

        <TextView
            android:layout_width="75dp"
            android:layout_height="75dp"
            android:layout_column="2"
            android:layout_row="3"
            android:background="#00ff00"
            android:gravity="center"
            android:text="three" />

        <TextView
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_column="0"
            android:layout_row="1"
            android:background="#0000ff"
            android:gravity="center"
            android:text="four" />

        <TextView
            android:layout_width="50dp"
            android:layout_height="200dp"
            android:layout_column="3"
            android:layout_row="0"
            android:layout_rowSpan="4"
            android:background="#0077ff"
            android:gravity="center"
            android:text="five" />

    </GridLayout>
</RelativeLayout>
```

Dynamic Definition

See the slides