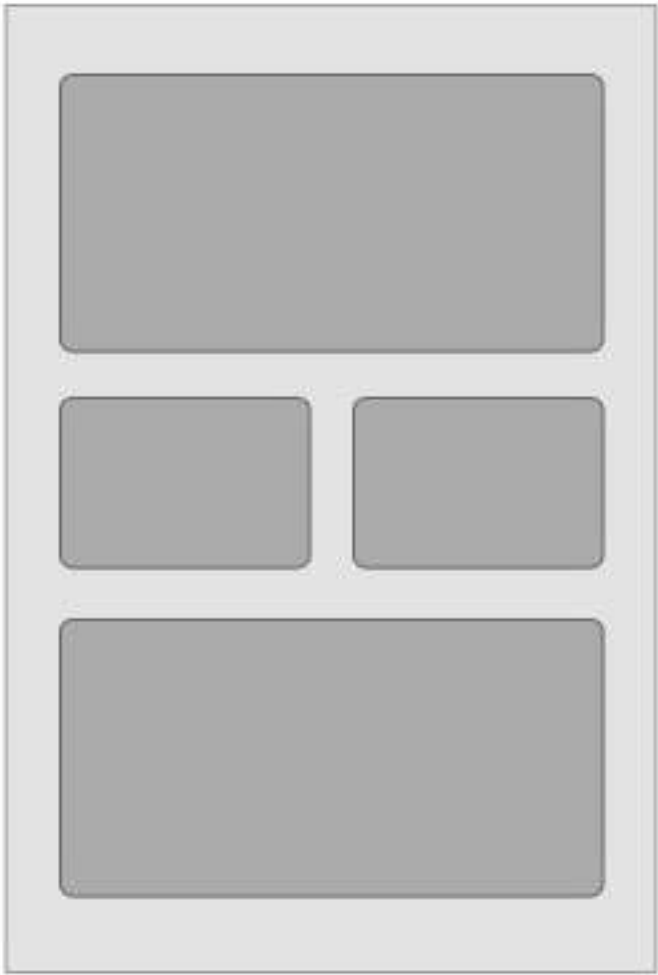


# Android Relative Layout

[Previous Page](#)

[Next Page](#)

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.



**Relative Layout**

## RelativeLayout Attributes

Following are the important attributes specific to RelativeLayout –

Sr.No.	Attribute & Description
1	<b>android:id</b> This is the ID which uniquely identifies the layout.

2	<b>android:gravity</b> This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
3	<b>android:ignoreGravity</b> This indicates what view should not be affected by gravity.

Using RelativeLayout, you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from **RelativeLayout.LayoutParams** and few of the important attributes are given below –

Sr.No.	Attribute & Description
1	<b>android:layout_above</b> Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name"
2	<b>android:layout_alignBottom</b> Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
3	<b>android:layout_alignLeft</b> Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
4	<b>android:layout_alignParentBottom</b> If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".
5	<b>android:layout_alignParentEnd</b> If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".

6	<b>android:layout_alignParentLeft</b>  If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".
7	<b>android:layout_alignParentRight</b>  If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".
8	<b>android:layout_alignParentStart</b>  If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".
9	<b>android:layout_alignParentTop</b>  If true, makes the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".
10	<b>android:layout_alignRight</b>  Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
11	<b>android:layout_alignStart</b>  Makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
12	<b>android:layout_alignTop</b>  Makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
13	<b>android:layout_below</b>  Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
14	<b>android:layout_centerHorizontal</b>  If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
15	<b>android:layout_centerInParent</b>

	If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".
16	<b>android:layout_centerVertical</b> If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
17	<b>android:layout_toEndOf</b> Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+] [package:]type:name".
18	<b>android:layout_toLeftOf</b> Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+] [package:]type:name".
19	<b>android:layout_toRightOf</b> Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+] [package:]type:name".
20	<b>android:layout_toStartOf</b> Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+] [package:]type:name".

## Example

This example will take you through simple steps to show how to create your own Android application using Relative Layout. Follow the following steps to modify the Android application we created in *Hello World Example* chapter –

Step	Description
1	You will use Android Studio IDE to create an Android application and name it as <i>demo</i> under a package <i>com.example.demo</i> as explained in the <i>Hello World Example</i> chapter.
2	Modify the default content of <i>res/layout/activity_main.xml</i> file to include few widgets in Relative layout.
3	Define required constants in <i>res/values/strings.xml</i> file

4

Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.demo/MainActivity.java**. This file can include each of the fundamental lifecycle methods.

```
package com.example.demo;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Following will be the content of **res/layout/activity\_main.xml** file –

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >

    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/name">
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button" />
```


```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button2" />
```

```
</LinearLayout>
```

```
</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants –

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="action_settings">Settings</string>
    <string name="reminder">Enter your name</string>
</resources>
```

Let's try to run our modified **Hello World!** application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android Studio, open one of your project's activity files and click Run  icon from the toolbar. Android Studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window –



[Previous Page](#)

[Print](#)

[Next Page](#)