ImageView Tutorial With Example In Android

In Android, [ImageView](https://abhiandroid.com/ui/imageview/) class is used to display an image file in application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices. An android is enriched with some of the best UI design widgets that allows us to build good looking and attractive UI based application.

**Important Note:** [ImageView](https://abhiandroid.com/ui/imageview/) comes with different configuration options to support different scale types. Scale type options are used for scaling the bounds of an image to the bounds of the [imageview](https://abhiandroid.com/ui/imageview/). Some of them scaleTypes configuration properties are center, center\_crop, fit\_xy, fitStart etc. You can read our [ScaleType tutorial](https://abhiandroid.com/ui/scaletype-imageview-example.html) to learn all details on it.

**Below is an ImageView code in XML:**

Make sure to save lion image in drawable folder

<ImageView

android:id="@+id/simpleImageView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:src="@drawable/lion" />

#### ****Attributes of ImageView:****

Now let’s  we discuss some important attributes that helps us to configure a ImageView in your [xml](https://abhiandroid.com/ui/xml/) file.

**1. id:** id is an attribute used to uniquely identify a [image view](https://abhiandroid.com/ui/imageview/) in android. Below is the example code in which we set the id of a [image view](https://abhiandroid.com/ui/imageview/).

<ImageView

**android:id="@+id/simpleImageView"**

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

/>

**2. src:** src is an attribute used to set a source file or you can say image in your imageview to make your layout attractive.

Below is the example code in which we set the source of a imageview lion which is saved in drawable folder.

<ImageView

android:id="@+id/simpleImageView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

**android:src="@drawable/lion"** /><!--set the source of an image view-->

**In Java:**

We can also set the source image at run time programmatically in [java](https://abhiandroid.com/java/) class. For that we use setImageResource() method as shown in below example code.

/\*Add in Oncreate() funtion after setContentView()\*/

ImageView simpleImageView=(ImageView) findViewById(R.id.*simpleImageView*);

simpleImageView.setImageResource(R.drawable.*lion*);//set the source in java class

**3. background:** background attribute is used to set the background of a ImageView. We can set a color or a drawable in the background of a ImageView.

Below is the example code in which we set the black color in the background and an image in the src attribute of [image view](https://abhiandroid.com/ui/imageview/).

<ImageView

android:id="@+id/simpleImageView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:src="@drawable/lion"

**android:background="#000"**/><!--black color in background of a image view-->

## ImageButton Tutorial With Example In Android Studio

In Android, [ImageButton](https://abhiandroid.com/ui/imagebutton/) is used to display a normal [button](https://abhiandroid.com/ui/button/) with a custom image in a [button](https://abhiandroid.com/ui/button). In simple words we can say, [ImageButton](https://abhiandroid.com/ui/imagebutton/) is a [button](https://abhiandroid.com/ui/button/) with an image that can be pressed or clicked by the users. By default it looks like a normal [button](https://abhiandroid.com/ui/button/) with the standard button background that changes the color during different button states.

An image on the surface of a button is defined within a [xml](https://abhiandroid.com/ui/xml/) (i.e. layout ) by using src attribute or within [java](https://abhiandroid.com/java/) class by using setImageResource() method. We can also set an image or custom drawable in the background of the [image button](https://abhiandroid.com/ui/imagebutton/).

**Important Note:** Standard button background image is displayed in the background of button whenever you create an [image button](https://abhiandroid.com/ui/imagebutton/). To remove that image, you can define your own background image in [xml](https://abhiandroid.com/ui/xml/) by using background attribute or in [java](https://abhiandroid.com/java/) class by using setBackground() method.

**Below is the code and image which shows how custom imagebutton looks in Android:**

**Important Note:**[ImageButton](https://abhiandroid.com/ui/imagebutton/) has all the properties of a normal button so you can easily perform any event like click or any other event which you can perform on a normal button.

**ImageButton code in XML:**

<!--Make Sure To Add Image Name home in Drawable Folder-->

<ImageButton

android:id="@+id/simpleImageButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/home" />

#### ****Attributes of ImageButton:****

Now let’s we discuss some important attributes that helps us to configure a [image button](https://abhiandroid.com/ui/imagebutton/) in your [xml](https://abhiandroid.com/ui/xml/) file (layout).

**1. id:** id is an attribute used to uniquely identify a image button. Below is the example code in which we set the id of a image button.

<ImageButton

**android:id="@+id/simpleImageButton"**

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

**2. src:** src is an attribute used to set a source file of image or you can say image in your image button to make your layout look attractive.

Below is the example code in which we set the source of an image button. Make sure you have saved an image in drawable folder name home before using below code.

<ImageButton

android:id="@+id/simpleImageButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

**android:src="@drawable/home"**/> <!--src(source)file from drawable folder which display an imagebutton-->

**Setting Image Source In ImageButton Using Java class:**

We can also set the source image at run time programmatically in [java](https://abhiandroid.com/java/) class. For that we use setImageResource() function as shown in below example code.

/\*Add in Oncreate() funtion after setContentView()\*/

ImageButton simpleImageButton = (ImageButton)findViewById(R.id.*simpleImageButton*);

simpleImageButton.setImageResource(R.drawable.*home*); //set the image programmatically

**3. background:** background attribute is used to set the background of an image button. We can set a color or a drawable in the background of a Button.

Below is the example code in which we set the black color for the background and an home image as the source of the image button.

<ImageButton

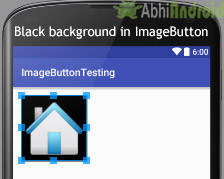
android:id="@+id/simpleImageButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/home"

android:background="#000"/><!-- black background color for image button-->



**Setting Background In ImageButton Using Java class:**

Below is the example code in which we set the black background color of a image button programmatically means in java class.

/\*Add in Oncreate() funtion after setContentView()\*/

ImageButton simpleImageButton = (ImageButton) findViewById(R.id.*simpleImageButton*);

simpleImageButton.setBackgroundColor(Color.*BLACK*); //set black background color for image button

**4. padding:** padding attribute is used to set the padding from left, right, top or bottom of the ImageButton.

* **paddingRight :** set the padding from the right side of the image button**.**
* **paddingLeft :** set the padding from the left side of the image button**.**
* **paddingTop :** set the padding from the top side of the image button**.**
* **paddingBottom :** set the padding from the bottom side of the image button**.**
* **padding :** set the padding from the all side’s of the image button**.**

Below is the example code of padding attribute in which we set the 20dp padding from all the side’s of a image button.

<ImageButton

android:id="@+id/simpleImageButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="#000"

android:src="@drawable/home"

android:padding="30dp"/><!-- set 30dp padding from all the sides of the view-->



## CheckBox Tutorial With Example In Android Studio

In Android, [CheckBox](https://abhiandroid.com/ui/checkbox/) is a type of two state [button](https://abhiandroid.com/ui/button/) either unchecked or checked in Android. Or you can say it is a type of on/off [switch](https://abhiandroid.com/ui/switch/) that can be toggled by the users. You should use [checkbox](https://abhiandroid.com/ui/checkbox/) when presenting a group of selectable options to users that are not mutually exclusive. CompoundButton is the parent class of [CheckBox](https://abhiandroid.com/ui/checkbox/) class.

n android there is a lot of usage of [check box](https://abhiandroid.com/ui/checkbox/). For example, to take survey in Android app we can list few options and allow user to choose using CheckBox. The user will simply checked these checkboxes rather than type their own option in [EditText](https://abhiandroid.com/ui/edittext). Another very common use of CheckBox is as remember me option in Login form.

**CheckBox code in XML:**

<CheckBox

android:id="@+id/simpleCheckBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Simple CheckBox"/>

#### ****Attributes of CheckBox:****

Now let’s  we discuss important attributes that helps us to configure a check box in [XML](https://abhiandroid.com/ui/xml/) file (layout).

**1.id:** id is an attribute used to uniquely identify a check box. Below we set the id of a image [button](https://abhiandroid.com/ui/button/).

<CheckBox

android:id="@+id/simpleCheckBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Simple CheckBox"/>

**2. checked:**checked is an attribute of check box used to set the current state of a check box. The value should be true or false where true shows the checked state and false shows unchecked state of a check box. The default value of checked attribute is false. We can also set the current state programmatically.

Below is an example code in which we set true value for checked attribute that sets the current state to checked.

<CheckBox

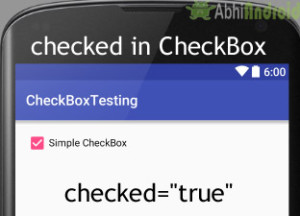
android:id="@+id/simpleCheckBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Simple CheckBox"

android:checked="true"/> <!--set the current state of the check box-->



**Setting Current State Of CheckBox In Java Class:**

Below we set the current state of CheckBox in [java](https://abhiandroid.com/java/) class.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a check box

CheckBox simpleCheckBox = (CheckBox) findViewById(R.id.*simpleCheckBox*);

// set the current state of a check box

simpleCheckBox.setChecked(true);

**3. gravity:**The gravity attribute is an optional attribute which is used to control the alignment of the text in CheckBox like left, right, center, top, bottom, center\_vertical, center\_horizontal etc.

Below we set the right and center\_vertical gravity for the text of a check box.

<CheckBox

android:id="@+id/simpleCheckBox"

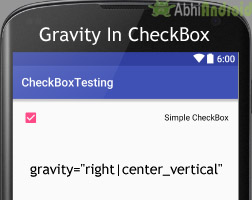
android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Simple CheckBox"

android:checked="true"

android:gravity="right|center\_vertical"/> <!-- gravity of the check box-->



**4. text:** text attribute is used to set the text in a check box. We can set the text in [xml](https://abhiandroid.com/ui/xml/) as well as in the [java](https://abhiandroid.com/java/) class.

Below is the example code with explanation included in which we set the text “Text Attribute Of Check Box” for a check box.

<CheckBox

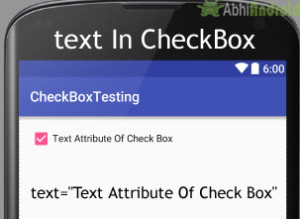
android:id="@+id/simpleCheckBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:text="Text Attribute Of Check Box"/> <!--displayed text of the check box-->



**Setting text in CheckBox In Java class:**

Below is the example code in which we set the text of a check box programmatically means in [java](https://abhiandroid.com/java/) class.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate check box

CheckBox simpleCheckBox = (CheckBox) findViewById(R.id.***simpleCheckBox***);

// displayed text of the check box

simpleCheckBox.setText(**"Text Attribute Of Check Box"**);

**5. textColor:** textColor attribute is used to set the text color of a check box. Color value is in form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below we set the red color for the displayed text of a check box.

<CheckBox

android:id="@+id/simpleCheckBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Text is Red Color"

android:textColor="#f00"

android:checked="true"/> <!-- red color for the text of check box-->

## Switch (On/Off) Tutorial With Example In Android Studio

In Android, [Switch](https://abhiandroid.com/ui/switch/) is a two-state toggle [switch](https://abhiandroid.com/ui/switch/) widget that can select between two options. It is used to display checked and unchecked state of a [button](https://abhiandroid.com/ui/button/) providing slider control to user. [Switch](https://abhiandroid.com/ui/switch/) is a subclass of CompoundButton. It is basically an off/on [button](https://abhiandroid.com/ui/button/) which indicate the current state of Switch. It is commonly used in selecting on/off in Sound, Bluetooth, WiFi etc.



As compared to [ToggleButton](https://abhiandroid.com/ui/togglebutton/) Switch provides user slider control. The user can simply tap on a switch to change its current state.

Switch allow the users to change the setting between two states like turn on/off  wifi, Bluetooth etc from your phone’s setting menu. It was introduced after Android 4.0 version (API level 14).

**Important Note:**Android Switch and [ToggleButton](https://abhiandroid.com/ui/togglebutton/) both are the subclasses of CompoundButton class.

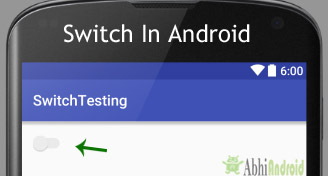
**Switch code in XML:**

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>



**Important Note:** We can check the current state of a Switch programmatically by using isChecked() method. This method returns a Boolean value means true or false. If a Switch is checked then it returns true otherwise it returns false.

Below is an example code in which we checked the current state of a Switch.

// initiate a Switch

Switch simpleSwitch = (Switch) findViewById(R.id.*simpleSwitch*);

// check current state of a Switch (true or false).

Boolean switchState = simpleSwitch.isChecked();

**Table Of Contents**[[hide](https://abhiandroid.com/ui/switch)]

* [1 Attributes of Switch:](https://abhiandroid.com/ui/switch#Attributes_of_Switch)
* [2 Example of Switch In Android Studio:](https://abhiandroid.com/ui/switch#Example_of_Switch_In_Android_Studio)

#### ****Attributes of Switch:****

Now let’s  we discuss important Switch attributes that helps us to configure a Switch in [XML](https://abhiandroid.com/ui/xml/) file(layout).

**1. id:** id is an attribute used to uniquely identify a Switch.

<Switch

**android:id="@+id/simpleSwitch"**

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

**2. checked:**checked attribute of Switch is used to set the current state of a Switch. The value can be either true or false where true shows the checked state and false shows unchecked state of a Switch. The default value of checked attribute is false. We can also set the current state programmatically.

Below we set “true” value for checked attribute that sets the current state to checked.

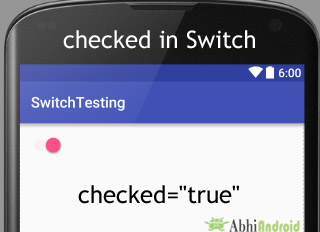
<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"/> <!-- set the current state of the Switch-->



**Setting Check Attribute In Switch Using Java Class:**

Below we set the current state of Switch in [java](https://abhiandroid.com/java/) class.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a Switch

Switch simpleSwitch = (Switch) findViewById(R.id.*simpleSwitch*);

//set the current state of a Switch

simpleSwitch.setChecked(true);

**3. text:** text attribute is used to set the text in a Switch. We can set the text in [xml](https://abhiandroid.com/ui/xml/) as well as in the [java](https://abhiandroid.com/java/) class.

Below we set the text “Sound” for the Switch.

<Switch

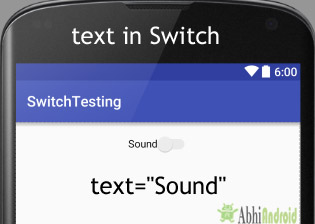
android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:text="Sound" /> <!--displayed text of switch-->



**Setting Text In Switch Using Java class:**

In the below code we set text of Switch via [java](https://abhiandroid.com/java/) class.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate Switch

Switch simpleSwitch = (Switch) findViewById(R.id.*simpleSwitch*);

//displayed text of the Switch

simpleSwitch.setText("switch");

**4. gravity:**The gravity attribute is an optional attribute which is used to control the alignment of the text in Switch. We can set text left, right, center, top, bottom, center\_vertical, center\_horizontal etc in Switch.

In the below code we set the left gravity for the text in Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Sound"

android:checked="true"

android:gravity="left"/><!--gravity of the Switch-->



**5. textOn And testOff:**textOn attribute is used to set the text when Switch is in checked state (i.e. on state). We can set the textOn in [XML](https://abhiandroid.com/ui/xml/) as well as in the java class.

In the below example we set the textOn as “Yes” and textOff as “No” for a Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

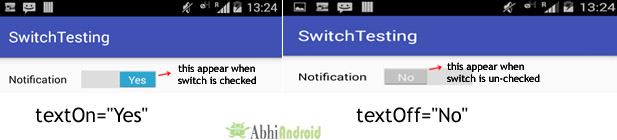
android:layout\_height="wrap\_content"

android:checked="true"

android:text="Notification"

android:textOff="No"

android:textOn="Yes"/><!-- text to be displayed whenever current state is checked-->



**Setting textOn And testOff of Switch In Java class:**

Below we set the textOn and textOff of a Switch in java class.

/\*Add in Oncreate() funtion after setContentView()\*/

Switch simpleSwitch = (Switch) findViewById(R.id.***simpleSwitch***); // initiate Switch

simpleSwitch.setTextOn(**"On"**); // displayed text of the Switch whenever it is in checked or on state

simpleSwitch.setTextOff("Off"); // displayed text of the Switch whenever it is in unchecked i.e. off state

**6. textColor:**textColor attribute is used to set the text color of a Switch. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below is the example code with explanation included in which we set the red color for the displayed text of a Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:text="switch"

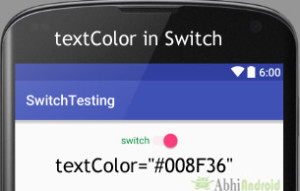
android:layout\_centerHorizontal="true"

android:textOn="On"

android:textOff="Off"

android:gravity="center"

android:textColor="#008F36"/><!-- Green color for displayed text-->



**Setting textColor Of Switch text In Java class:**

Below is the example code in which we set the text color of a Switch programmatically.

/\*Add in Oncreate() funtion after setContentView()\*/

Switch simpleSwitch = (Switch) findViewById(R.id.simpleSwitch);// initiate Switch

simpleSwitch.setTextColor(Color.GREEN); //red color for displayed text of Switch

**7. textSize:** textSize attribute is used to set the size of the text of a Switch. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below is the example code in which we set the 25sp size for the text of a Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:text="switch"

android:layout\_centerHorizontal="true"

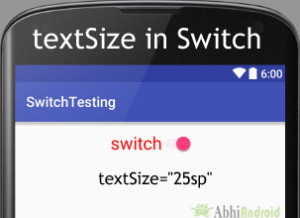
android:textOn="On"

android:textOff="Off"

android:textColor="#f00"

android:gravity="center"

android:textSize="25sp"/> <!--25sp displayed text size-->



**Setting textSize Of Switch Text In Java class:**

Below is the example code in which we set the text size of a Switch programmatically.

Switch simpleSwitch = (Switch) findViewById(R.id.***simpleSwitch***); // initiate Switch

simpleSwitch.setTextSize(25); // set 25sp displayed text size of Switch

**9. textStyle:** textStyle attribute is used to set the text style of the text of a Switch. The possible text styles are bold, italic and normal.  If we need to use two or more styles for a [text view](https://abhiandroid.com/ui/textview/) then use “|” operator.

Below we set the bold and italic text styles for text of a Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="false"

android:text="switch"

android:layout\_centerHorizontal="true"

android:textOn="On"

android:textOff="Off"

android:textColor="#f00"

android:gravity="center"

android:textSize="25sp"

android:textStyle="bold|italic"/><!--bold and italic text style for displayed text-->



**10. background:** background attribute is used to set the background of a Switch. We can set a color or a drawable in the background of a Switch.

Below we set the black color for the background and red color for the displayed text of a Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:text="switch"

android:layout\_centerHorizontal="true"

android:textOn="On"

android:textOff="Off"

android:textColor="#f00"

android:padding="20dp"

android:gravity="center"

android:textSize="25sp"

android:background="#000"/><!-- black background color-->



**Setting Background In Java class:**

Below we set the background color of a Switch programmatically.

Switch simpleSwitch = (Switch) findViewById(R.id.*simpleSwitch*);

simpleSwitch.setBackgroundColor(Color.*BLACK*);

**11. padding:** padding attribute is used to set the padding from left, right, top or bottom in Switch.

* **paddingRight:** set the padding from the right side of the Switch**.**
* **paddingLeft:** set the padding from the left side of the Switch**.**
* **paddingTop:** set the padding from the top side of the Switch**.**
* **paddingBottom:** set the padding from the bottom side of the Switch**.**
* **Padding:** set the padding from the all side’s of the Switch**.**

Below we set the 20dp padding from all the side’s of the Switch.

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:text="switch"

android:layout\_centerHorizontal="true"

android:textOn="On"

android:textOff="Off"

android:textColor="#f00"

android:gravity="center"

android:textSize="25sp"

android:padding="40dp"/><!-- 20dp padding from all the side's-->



**12. drawableBottom, drawableTop, drawableRight And drawableLeft:** These attribute draw the drawable below, top, right and left of the text of Switch.

Below we set the icon to the below of the text of a Switch. You can try other three by yourself.

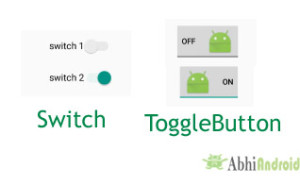
## ToggleButton (On/Off) Tutorial With Example In Android

In Android, [ToggleButton](https://abhiandroid.com/ui/togglebutton/) is used to display checked and unchecked state of a [button](https://abhiandroid.com/ui/button/). [ToggleButton](https://abhiandroid.com/ui/togglebutton/) basically an off/on [button](https://abhiandroid.com/ui/button/) with a light indicator which indicate the current state of toggle [button](https://abhiandroid.com/ui/button/). The most simple example of [ToggleButton](https://abhiandroid.com/ui/togglebutton/) is doing on/off in sound, Bluetooth, wifi, hotspot etc. It is a subclass of compoundButton.



**ToggleButton Vs Switch In Android:**

ToggleButton allow the users to change the setting between two states like turn on/off your wifi, Bluetooth etc from your phone’s setting menu. Since, Android 4.0 version ( API level 14 ) there is an another kind of ToggleButton called [Switch](https://abhiandroid.com/ui/switch/) which provide the user slider control. You can learn more about it reading [Switch tutorial](https://abhiandroid.com/ui/switch).



**Important Note:**Android [Switch](https://abhiandroid.com/ui/switch/) and ToggleButton both are the subclasses of CompoundButton class.

**ToggleButton code in XML:**

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

**How To Check Current State Of ToggleButton:**

To check current state of a [toggle button](https://abhiandroid.com/ui/togglebutton/) programmatically we use isChecked() method. This method returns a Boolean value either true or false. If a [toggle button](https://abhiandroid.com/ui/togglebutton/) is checked then it returns true otherwise it returns false. Below is the code which checks the current state of a [toggle button](https://abhiandroid.com/ui/togglebutton/).

/\*Add in Oncreate() funtion after setContentView()\*/

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleButton); // initiate a toggle button

Boolean ToggleButtonState = simpleToggleButton.isChecked(); // check current state of a toggle button (true or false).

**Table Of Contents**[[hide](https://abhiandroid.com/ui/togglebutton)]

* [1 Attributes of ToggleButton:](https://abhiandroid.com/ui/togglebutton#Attributes_of_ToggleButton)
* [2 ToggleButton Example In Android Studio:](https://abhiandroid.com/ui/togglebutton#ToggleButton_Example_In_Android_Studio)

#### ****Attributes of ToggleButton:****

Now let’s  we discuss important attributes that helps us to configure a Toggle Button in [XML](https://abhiandroid.com/ui/xml/) file (layout).

**1. id:** id is an attribute used to uniquely identify a toggle button.

<ToggleButton

**android:id="@+id/simpleToggleButton"**

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

**2. checked:**checked is an attribute of toggle button used to set the current state of a toggle button. The value should be true or false where true shows the checked state and false shows unchecked state of a toggle button. The default value of checked attribute is false. We can also set the current state programmatically.

Below we set true value for checked attribute sets the current state to checked.

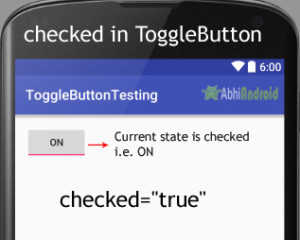
<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true" /><!-- set the current state of the toggle button-->



**Setting Checked Of ToogleButton Current State In Java Class:**

Below we set the current state of toggle button to checked:

/\*Add in Oncreate() funtion after setContentView()\*/

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.***simpleToggleButton***); // initiate a toggle button

simpleToggleButton.setChecked(**true**); // set the current state of a toggle button

**3. gravity:**The gravity attribute is an optional attribute which is used to control the alignment of the text in ToogleButton like left, right, center, top, bottom, center\_vertical, center\_horizontal etc.

<ToggleButton

android:id="@+id/simpleToggleButton"

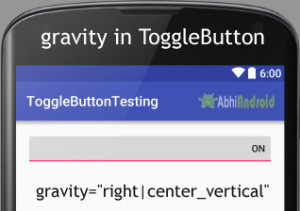
android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:checked="true"

android:gravity="right|center\_vertical"/><!-- gravity of the toggle button-->



**4. textOn And textOff:** textOn attribute is used to set the text when toggle button is in checked/on state. We can set the textOn in [XML](https://abhiandroid.com/ui/xml/) as well as in the [java](https://abhiandroid.com/java/) class.

Below is the example code with explanation included in which we set the textOn  “Enabble Attribute Of Toggle button” for a toggle button.

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

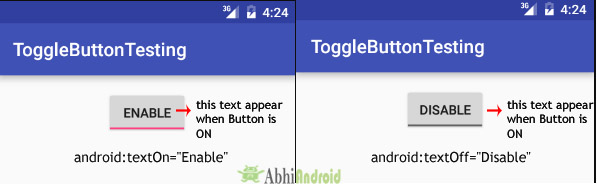
android:layout\_height="wrap\_content"

android:checked="true"

android:layout\_centerHorizontal="true"

android:textOff="Disable"

android:textOn="Enable"/> <!--text to be displayed whenever toggle button is checked-->



**Setting textOn and textOff Of ToggleButton In Java class:**

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate toggle button

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleButton);

// displayed text of the toggle button whenever it is in checked or on state

simpleToggleButton.setTextOn("TextOn Attribute Of Toggle b3`utton");

// displayed text of the toggle button whenever it is in unchecked or off state

simpleToggleButton.setTextOff("TextOff Attribute Of Toggle b3`utton");

**5. textColor:** textColor attribute is used to set the text color of a toggle button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below we set the red color for the displayed text of a Toggle button.

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="false"

android:textOff="On State"

android:textOn="Off State"

android:layout\_centerHorizontal="true"

android:textColor="#f00" /><!--red color for displayed text-->



**Setting textColor Of ToggleButton In Java class:**

/\*Add in Oncreate() funtion after setContentView()\*/

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.***simpleToggleButton***);// initiate toggle button

simpleToggleButton.setTextColor(Color.***RED***); //red color for displayed text of toggle button

**6. textSize:** textSize attribute set the size of the text of a toggle button. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below we set the 25sp size for the text of a toggle button.

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="false"

android:textOff="On State"

android:textOn="Off State"

android:layout\_centerHorizontal="true"

android:textColor="#f00"

android:textSize="25sp"/> <!-- 25sp displayed text size-->



**Setting textSize Of ToggleButton Text In Java class:**

/\*Add in Oncreate() funtion after setContentView()\*/

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.***simpleToggleButton***); // initiate toggle button

simpleToggleButton.setTextSize(25); // set 25sp displayed text size of toggle button

**7. textStyle:** textStyle attribute is used to set the text style of the text of a Toggle button. You can set bold, italic and normal.  If you need to use two or more styles for a [text view](https://abhiandroid.com/ui/textview/) then “|” operator is used for that.

Below we set the bold and italic text styles for text of a toggle button.

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textOff="Off State"

android:textOn="On State"

android:textSize="25sp"

android:layout\_centerHorizontal="true"

android:textColor="#f00"

android:textStyle="bold|italic"/> <!-- bold and italic text style for displayed text-->



**8. background:** background attribute is used to set the background of a toggle button. We can set a color or a drawable in the background of a toggle button.

Below we set the black color for the background and red color for the displayed text of a ToggleButton.

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textOff="Off State"

android:textOn="On State"

android:textSize="25sp"

android:layout\_centerHorizontal="true"

android:textStyle="bold|italic"

android:textColor="#f00"

android:background="#000"/><!--Black Color Background-->



**Setting Background Of ToggleButton Text In Java class:**

/\*Add in Oncreate() funtion after setContentView()\*/

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.***simpleToggleButton***);

simpleToggleButton.setBackgroundColor(Color.***BLACK***);

**9. padding:** padding attribute is used to set the padding from left, right, top or bottom.

* **paddingRight :**set the padding from the right side of the toggle button**.**
* **paddingLeft :**set the padding from the left side of the toggle button**.**
* **paddingTop :**set the padding from the top side of the toggle button**.**
* **paddingBottom :**set the padding from the bottom side of the toggle button**.**
* **Padding :**set the padding from the all side’s of the toggle button**.**

Below we set the 40dp padding from all the side’s of the toggle button.

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textOff="Off State"

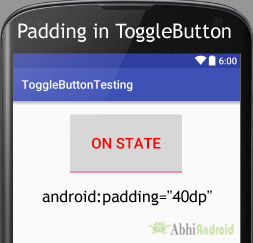
android:textOn="On State"

android:textSize="25sp"

android:layout\_centerHorizontal="true"

android:textColor="#f00"

android:padding="40dp"/> <!--40dp padding from all the side's of toggle button-->



**10. drawableBottom, drawableTop, drawableRight And drawableLeft:**These attribute draw the drawable below, top, right and left of the text of ToggleButton.

Below we set the icon to the Top of the text of a ToggleButton. In the similar way you can try for other three attribute yourself.

<!--Make sure to add ic\_launcher image in drawable folder-->

<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textOff="Off State"

android:textOn="On State"

android:layout\_centerHorizontal="true"

android:textColor="#000"

android:drawableTop="@drawable/ic\_launcher" /><!--drawable icon at the bottom of text of top buttton-->



#### ****ToggleButton Example In Android Studio:****

Below is the example of ToggleButton in Android Studio. In this example we display two toggle button with background and one “submit” button using attributes discussed earlier in this post. Whenever user click on the submit button, the current state of both toggle button’s is displayed in a Toast. Below is the final output, download code and step by step explanation:

[Download Code](https://github.com/abhisheksaini4/ToggleButtonExample)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) and name it ToggleButtonExample

In this step we [create a new project](https://abhiandroid.com/androidstudio/start-create-project) for ToggleButton in [Android Studio](https://abhiandroid.com/androidstudio/)by filling all the necessary details of the app like app name, package name, api versions etc.

Select File -> New -> New Project -> Fill the forms and click "Finish" button.

**Step 2:** Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml/) (or) main.xml and add following code:

In this step we open an xml file and add the code for displaying two toggle button and one normal Button.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:orientation="horizontal">

<ToggleButton

android:id="@+id/simpleToggleButton1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:checked="false"

android:drawablePadding="20dp"

android:drawableRight="@drawable/ic\_launcher"

android:textColor="#000" />

<ToggleButton

android:id="@+id/simpleToggleButton2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginLeft="50dp"

android:checked="true"

android:drawableLeft="@drawable/ic\_launcher"

android:drawablePadding="20dp"

android:textColor="#000" />

</LinearLayout>

<Button

android:id="@+id/submitButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:layout\_marginTop="50dp"

android:background="#0f0"

android:padding="10dp"

android:text="Submit"

android:textColor="#fff"

android:textSize="20sp"

android:textStyle="bold" />

</LinearLayout>

Step 3: Open   app -> [java](https://abhiandroid.com/java/) -> package -> **MainActivity.java**

In this step we open MainActivity where we add the code to initiate the Toggle Buttons and normal Button. After initiating we perform click event on button and display the text of current state of ToggleButton using a Toast.

package example.abhiandriod.togglebuttonexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

ToggleButton simpleToggleButton1, simpleToggleButton2;

Button submit;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// initiate toggle button's

simpleToggleButton1 = (ToggleButton) findViewById(R.id.simpleToggleButton1);

simpleToggleButton2 = (ToggleButton) findViewById(R.id.simpleToggleButton2);

submit = (Button) findViewById(R.id.submitButton);

submit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String status = "ToggleButton1 : " + simpleToggleButton1.getText() + "\n" + "ToggleButton2 : " + simpleToggleButton2.getText();

Toast.makeText(getApplicationContext(), status, Toast.LENGTH\_SHORT).show(); // display the current state of toggle button's

}

});

}

}

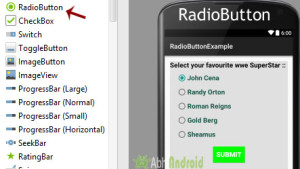
**Output:**

Now start the AVD in Emulator and run the App. You will see two ToggleButton and submit Button. Click on the submit button which will display the state of ToggleButton.



## RadioButton & RadioGroup Tutorial With Example In Android Studio

In Android, [RadioButton](https://abhiandroid.com/ui/radiobutton/) are mainly used together in a [RadioGroup](https://abhiandroid.com/ui/radiobutton/). In [RadioGroup](https://abhiandroid.com/ui/radiobutton/) checking the one radio [button](https://abhiandroid.com/ui/button/) out of several radio [button](https://abhiandroid.com/ui/button/) added in it will automatically unchecked all the others. It means at one time we can checked only one radio [button](https://abhiandroid.com/ui/button/) from a group of radio buttons which belong to same [radio group](https://abhiandroid.com/ui/radiobutton/). The most common use of [radio button](https://abhiandroid.com/ui/radiobutton/) is in [Quiz Android App code](https://abhiandroid.com/sourcecode/quiz).



RadioButon is a two state button that can be checked or unchecked. If a [radio button](https://abhiandroid.com/ui/radiobutton/) is unchecked then a user can check it by simply clicking on it. Once a RadiaButton is checked by user it can’t be unchecked by simply pressing on the same button. It will automatically unchecked when you press any other [RadioButton](https://abhiandroid.com/ui/radiobutton/) within same [RadioGroup](https://abhiandroid.com/ui/radiobutton/).

**Important Note:**RadioGroup is a widget used in Android for the grouping of radio buttons and provide the feature of selecting only one [radio button](https://abhiandroid.com/ui/radiobutton/) from the set. When a user try to select any other radio button within same [radio group](https://abhiandroid.com/ui/radiobutton/) the previously selected radio button will be automatically unchecked.

**RadioGroup And RadioButton code in XML:**

<RadioGroup

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

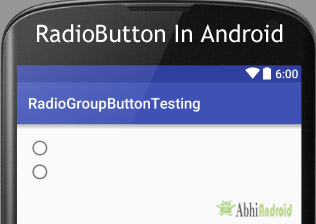
<RadioButton

android:id="@+id/simpleRadioButton1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

</RadioGroup>



**Checking Current State Of Radio Button:**

You can check the current state of a radio button programmatically by using isChecked() method. This method returns a Boolean value either true or false. if it is checked then returns true otherwise returns false. Below is an example code with explanation in which we checked the current state of a radio button.

/\*Add in Oncreate() funtion after setContentView()\*/

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.***simpleRadioButton***); // initiate a radio button

Boolean RadioButtonState = simpleRadioButton.isChecked(); // check current state of a radio button (true or false).

**Table Of Contents**[[hide](https://abhiandroid.com/ui/radiobutton)]

* [1 Attributes of RadioButton In Android:](https://abhiandroid.com/ui/radiobutton#Attributes_of_RadioButton_In_Android)
* [2 Example Of RadioButton And RadioGroup in Android Studio:](https://abhiandroid.com/ui/radiobutton#Example_Of_RadioButton_And_RadioGroup_in_Android_Studio)

#### ****Attributes of RadioButton In Android:****

Now let’s  we discuss important attributes that helps us to create a beautiful radio button in [xml](https://abhiandroid.com/ui/xml/) file (layout).

**1. id:**id is an attribute used to uniquely identify a radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

**2. checked:**checked attribute in radio button is used to set the current state of a radio button. We can set it either true or false where true shows the checked state and false shows unchecked state of a radio button. As usual default value of checked attribute is false. We can also set the current state in [JAVA](https://abhiandroid.com/java/).

Below we set true value for checked attribute which sets the current state to checked of a Button

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"/> <!-- set the current state of the radio button-->



**Setting checked State Of RadioButton In Java Class:**

Below code set the current state of [RadioButton](https://abhiandroid.com/ui/radiobutton/) to checked programmatically.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a radio button

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.***simpleRadioButton***);

// set the current state of a radio button

simpleRadioButton.setChecked(**true**);

**3. text:** text attribute is used to set the text in a radio button. We can set the text both ways either in [XML](https://abhiandroid.com/ui/xml/) or in [JAVA](https://abhiandroid.com/java/) class.

Below is the example code with explanation included in which we set the text “I am a radiobutton” of a  radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:layout\_centerHorizontal="true"

android:text="I am a radiobutton" /> <!-- displayed text of radio button-->

**Setting text of RadioButton In Java class:**

Below we set the text of a radio button programmatically:

/\*Add in Oncreate() funtion after setContentView()\*/

RadioButton simpleRadioButton=(RadioButton) findViewById(R.id.simpleRadioButton);

simpleRadioButton.setText("I am a radiobutton"); // displayed text of radio button



**4. gravity:**The gravity attribute is an optional attribute which is used to control the alignment of text like left, right, center, top, bottom, center\_vertical, center\_horizontal etc.

Below is the example code with explanation included in which we set the center gravity for the text of a radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:checked="true"

android:text="I am a Button"

android:gravity="center"/> <!-- center gravity of the text-->



**5. textColor:** textColor attribute is used to set the text color of a radio button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below we set the red color for the displayed text of a radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:layout\_centerHorizontal="true"

android:text="Male"

android:textColor="#f00" /><!--red color for displayed text-->



**Setting textColor of RadioButton text In Java class:**

Below we set the text color of a radio button programmatically.

/\*Add in Oncreate() funtion after setContentView()\*/

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.***simpleRadioButton***);// initiate radio button

simpleRadioButton.setTextColor(Color.***RED***); //red color for displayed text of radio button

**6. textSize:** textSize attribute is used to set the size of the text of a radio button. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below we set the 25sp size for the text of a radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:layout\_centerHorizontal="true"

android:text="AbhiAndroid"

android:textColor="#f00"

android:textSize="25sp"/> <!--setting text size-->



**Setting textSize Of RadioButton Text In Java class:**

Below we set the text size of a radio button programmatically:

/\*Add in Oncreate() funtion after setContentView()\*/

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.***simpleRadioButton***); // initiate radio button

simpleRadioButton.setTextSize(25); // set 25sp displayed text size of radio button

**7. textStyle:** textStyle attribute is used to set the text style of the text of a radio button. The possible text styles are bold, italic and normal. If we need to use two or more styles for a [text view](https://abhiandroid.com/ui/textview/) then “|” operator is used for that.

Below is the example code with explanation included in which we set the bold and italic text styles for text of a radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textSize="25sp"

android:layout\_centerHorizontal="true"

android:text="Male"

android:textColor="#f00"

android:textStyle="bold|italic"/> <!-- bold and italic text style-->



**8. background:** background attribute is used to set the background of a radio button. We can set a color or a drawable in the background of a radio button.

Below we set the black color for the background and red color for the displayed text of a radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textSize="25sp"

android:textStyle="bold|italic"

android:padding="20dp"

android:layout\_centerHorizontal="true"

android:text="Male"

android:textColor="#f00"

android:background="#000"/> <!-- black background for radio button-->



**Setting Background Of RadioButton In Java class:**

Below we set the background color of a radio button programmatically.

/\*Add in Oncreate() funtion after setContentView()\*/

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.***simpleRadioButton***);

simpleRadioButton.setBackgroundColor(Color.***BLACK***);

**9. padding:** padding attribute is used to set the padding from left, right, top or bottom.

* **paddingRight:** set the padding from the right side of the radio button**.**
* **paddingLeft :** set the padding from the left side of the radio button**.**
* **paddingTop :** set the padding from the top side of the radio button**.**
* **paddingBottom:** set the padding from the bottom side of the radio button**.**
* **Padding:** set the padding from the all side’s of the radio button**.**

Below we set padding attribute of 20dp padding from all the side’s of the radio button.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textSize="25sp"

android:textStyle="bold|italic"

android:layout\_centerHorizontal="true"

android:text="AbhiAndroid"

android:textColor="#f00"

android:padding="40dp"/> <!--40dp padding from all the sides of radio button-->



**10. drawableBottom, drawableTop, drawableLeft And drawableRight:** These attribute draw the drawable to the below of the text of RadioButton.

Below we set the icon to the right of the text of a RadioButton.

<RadioButton

android:id="@+id/simpleRadioButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true"

android:textSize="25sp"

android:padding="20dp"

android:layout\_centerHorizontal="true"

android:text="AbhiAndroid"

android:textColor="#f00"

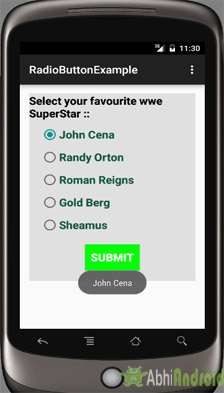
android:drawableRight="@drawable/ic\_launcher" /> <!-- drawable icon at the right of radio button-->



#### ****Example Of RadioButton And RadioGroup in Android Studio:****

Below is the example of Radiobutton in Android where we display five radio buttons with background and other attributes. The radio buttons are used to select your favorite WWE superstar with one “submit” button. Below is the final output, download code and step by step explanation of tutorial:

[Download Code](https://github.com/abhisheksaini4/RadioButtonExample)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) and name it RadioButtonExample

Select File -> New -> New Project and Fill the forms and click “Finish” button.

**Step 2:** Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml/) (or) main.xml and add following code:

In this step we open an xml file and add the code for displaying 5 RadioButton and one normal button.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:background="#e0e0e0"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Select your favourite wwe SuperStar :: "

android:textColor="#000"

android:textSize="20sp"

android:textStyle="bold" />

<RadioGroup

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<RadioButton

android:id="@+id/johnCena"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="10dp"

android:checked="true"

android:text="@string/johnCena"

android:textColor="#154"

android:textSize="20sp"

android:textStyle="bold" />

<RadioButton

android:id="@+id/randyOrton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="10dp"

android:checked="false"

android:text="@string/randyOrton"

android:textColor="#154"

android:textSize="20sp"

android:textStyle="bold" />

<RadioButton

android:id="@+id/romanReigns"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="10dp"

android:checked="false"

android:text="@string/romanReigns"

android:textColor="#154"

android:textSize="20sp"

android:textStyle="bold" />

<RadioButton

android:id="@+id/goldBerg"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="10dp"

android:checked="false"

android:text="@string/goldBerg"

android:textColor="#154"

android:textSize="20sp"

android:textStyle="bold" />

<RadioButton

android:id="@+id/sheamus"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="10dp"

android:checked="false"

android:text="@string/sheamus"

android:textColor="#154"

android:textSize="20sp"

android:textStyle="bold" />

</RadioGroup>

<Button

android:id="@+id/submitButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:layout\_margin="20dp"

android:background="#0f0"

android:padding="10dp"

android:text="Submit"

android:textColor="#fff"

android:textSize="20sp"

android:textStyle="bold" />

</LinearLayout>

</LinearLayout>

**Step 3:** Open  src -> package -> **MainActivity.java**

In this step we open MainActivity and add the code to initiate the RadioButton and normal button. We also perform click event on button and display the selected superstar’s name by using a Toast.

package example.gb.radiobuttonexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.RadioButton;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

RadioButton johnCena, randyOrton, goldBerg, romanReigns, sheamus;

String selectedSuperStar;

Button submit;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

johnCena = (RadioButton) findViewById(R.id.johnCena);

randyOrton = (RadioButton) findViewById(R.id.randyOrton);

goldBerg = (RadioButton) findViewById(R.id.goldBerg);

romanReigns = (RadioButton) findViewById(R.id.romanReigns);

sheamus = (RadioButton) findViewById(R.id.sheamus);

submit = (Button) findViewById(R.id.submitButton);

submit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if (randyOrton.isChecked()) {

selectedSuperStar = randyOrton.getText().toString();

} else if (sheamus.isChecked()) {

selectedSuperStar = sheamus.getText().toString();

} else if (johnCena.isChecked()) {

selectedSuperStar = johnCena.getText().toString();

} else if (romanReigns.isChecked()) {

selectedSuperStar = romanReigns.getText().toString();

} else if (goldBerg.isChecked()) {

selectedSuperStar = goldBerg.getText().toString();

}

Toast.makeText(getApplicationContext(), selectedSuperStar, Toast.LENGTH\_LONG).show(); // print the value of selected super star

}

});

}

}

**Step 4:** Open res -> values -> **strings.xml**

In this step we open String file which is used to store string data of the app.

<resources>

<string name="app\_name">RadioButtonExample</string>

<string name="hello\_world">Hello world!</string>

<string name="action\_settings">Settings</string>

<string name="randyOrton">Randy Orton</string>

<string name="johnCena">John Cena</string>

<string name="romanReigns">Roman Reigns</string>

<string name="goldBerg">Gold Berg</string>

<string name="sheamus">Sheamus</string>

</resources>

## RatingBar Tutorial With Example In Android Studio

[RatingBar](https://abhiandroid.com/ui/ratingbar/) is used to get the rating from the app user. A user can simply touch, drag or click on the stars to set the rating value. The value of rating always returns a floating point number which may be 1.0, 2.5, 4.5 etc.



In Android, [RatingBar](https://abhiandroid.com/ui/ratingbar/) is an extension of [ProgressBar](https://abhiandroid.com/ui/progressbar/) and [SeekBar](https://abhiandroid.com/ui/seekbar/) which shows a rating in stars. [RatingBar](https://abhiandroid.com/ui/ratingbar/) is a subclass of AbsSeekBar class.

The getRating() method of android RatingBar class returns the rating number.

**RatingBar code in XML:**

<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

**Table Of Contents**[[hide](https://abhiandroid.com/ui/ratingbar)]

* [1 Methods Used in Rating Bar:](https://abhiandroid.com/ui/ratingbar#Methods_Used_in_Rating_Bar)
* [2 Attributes Used in RatingBar:](https://abhiandroid.com/ui/ratingbar#Attributes_Used_inRatingBar)
* [3 RatingBar Example In Android Studio:](https://abhiandroid.com/ui/ratingbar#RatingBar_Example_In_Android_Studio)
* [4 Custom RatingBar Example In Android Studio:](https://abhiandroid.com/ui/ratingbar#Custom_RatingBar_Example_In_Android_Studio)

#### ****Methods Used in Rating Bar:****

**getRating():**

You can get the rating number from a RatingBar by using getRating() method. This method returns a Floating Point number. Below we get the current rating number from a RatingBar.

/\*Add in Oncreate() funtion after setContentView()\*/

RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.*simpleRatingBar*); // initiate a rating bar

Float ratingNumber = simpleRatingBar.getRating(); // get rating number from a rating bar

**getNumStars():**

You can get the number of stars of a RatingBar by using getNumstars() method. This method returns int value. In below code we get the total number of stars of a RatingBar.

/\*Add in Oncreate() funtion after setContentView()\*/

RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.*simpleRatingBar*); // initiate a rating bar

int numberOfStars = simpleRatingBar.getNumStars(); // get total number of stars of rating bar

#### ****Attributes Used in RatingBar:****

Now let’s we discuss some important attribute that helps us to configure a RatingBar in [XML](https://abhiandroid.com/ui/xml/) file (layout).

**1. id:**id is an attribute used to uniquely identify a [rating bar](https://abhiandroid.com/ui/ratingbar/).

<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

**2. background:** background attribute is used to set the background of a RatingBar. We can set a color or a drawable in the background of a [rating bar](https://abhiandroid.com/ui/ratingbar/).

Below we set the red color for the background of a [rating bar](https://abhiandroid.com/ui/ratingbar/).

<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="#f00"/> <!-- red color for the background of rating bar-->



**Setting Background of RatingBar In Java class:**

RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.*simpleRatingBar*); // initiate a rating bar

simpleRatingBar.setBackgroundColor(Color.*RED*); // set background color for a rating bar

**3. numStars:** numStars attribute is used to set the number of stars (or rating items) to be displayed in a rating bar. By default a rating bar shows five stars but we can change it using numStars attribute.

numStars must have a integer number like 1,2 etc.

Below we set num stars value to 7 of RatingBar.

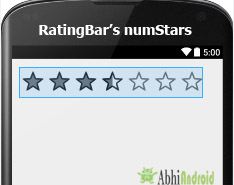
<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:numStars="7" /><!-- number of stars to be displayed-->



**Setting numStars of RatingBar In Java class:**

RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.*simpleRatingBar*); // initiate a rating bar

simpleRatingBar.setNumStars(7); // set total number of stars

**4. rating:** Rating attribute set the default rating of a rating bar. It must be a floating point number.

Below we set default rating to 3.5 for a rating bar.

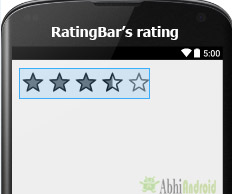
<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:rating="3.5" /> <!-- default rating-->



**Setting Default Rating of RatingBar In Java class:**

/\*Add in Oncreate() funtion after setContentView()\*/

RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.*simpleRatingBar*); // initiate a rating bar

simpleRatingBar.setRating((float) 3.5); // set default rating

**5. padding:** padding attribute is used to set the padding from left, right, top or bottom.

* **paddingRight:** set padding from the right side of the rating bar**.**
* **paddingLeft:** set padding from the left side of the rating bar**.**
* **paddingTop:** set padding from the top side of the rating bar**.**
* **paddingBottom:** set the padding from the bottom side of the rating bar**.**
* **Padding:** set the padding from the all side’s of the rating bar**.**

Below we set the 20dp padding from all the side’s of the rating bar.

<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:rating="2.5"

android:numStars="6"

android:background="#f00"

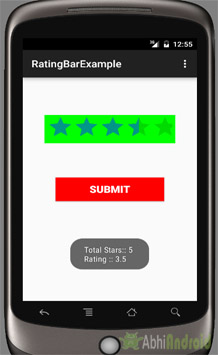
android:padding="20dp"/> <!--20dp padding from all the sides of rating bar-->



#### ****RatingBar Example In Android Studio:****

Below is the example of RatingBar in Android where we displayed a RatingBar with five stars and a submit [button](https://abhiandroid.com/ui/button/). Whenever a user click on the [button](https://abhiandroid.com/ui/button/) value of total number of stars and value of rating is shown by using a Toast on screen. Below is the final output, download code and step by step tutorial:

[Download Code](https://github.com/abhisheksaini4/RatingBarExample)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Important Note:** We can also create custom rating bar in Android in which we can change the star images like filled or empty star. We discussed this in next example, so don’t miss it.

**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) and name it RatingBarExample

Select File -> New -> New Project and Fill the forms and click "Finish" button.

**Step 2:** Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml/) (or) main.[xml](https://abhiandroid.com/ui/xml/) and add following code

In this step we open an xml file and add the code for displaying a rating bar with five number of stars and “2” value for default rating and one submit [button](https://abhiandroid.com/ui/button/).

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<RatingBar

android:id="@+id/simpleRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="60dp"

android:background="#0f0"

android:paddingLeft="5dp"

android:paddingRight="5dp"

android:rating="2" />

<Button

android:id="@+id/submitButton"

android:layout\_width="200dp"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:background="#f00"

android:padding="10dp"

android:text="Submit"

android:textColor="#fff"

android:textSize="20sp"

android:textStyle="bold" />

</RelativeLayout>

**Step 3:** Open src -> package -> **MainActivity.java**

In this step we open MainActivity where we add the code to initiate the RatingBar & button and then we perform click event on button and display the total number of stars and rating by using a toast.

package example.gb.ratingbarexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.RatingBar;

import android.widget.Button;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// initiate rating bar and a button

final RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.simpleRatingBar);

Button submitButton = (Button) findViewById(R.id.submitButton);

// perform click event on button

submitButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// get values and then displayed in a toast

String totalStars = "Total Stars:: " + simpleRatingBar.getNumStars();

String rating = "Rating :: " + simpleRatingBar.getRating();

Toast.makeText(getApplicationContext(), totalStars + "\n" + rating, Toast.LENGTH\_LONG).show();

}

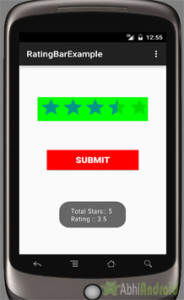
});

}

}

**Output:**

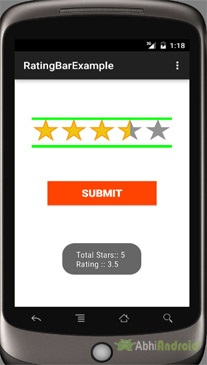
Now start the AVD in Emulator and run the App. You will see the rating option on screen. Select your rating and click on Submit. Your rating will be displayed on screen as Toast.



#### ****Custom RatingBar Example In Android Studio:****

In the below example of custom rating bar in Android, we displayed a rating bar with five **custom stars** and a submit button. Whenever a user click on the button value of total number of stars and value of rating is shown by using a Toast. Below is the final output, download code and step by step explanation of Custom RatingBar.

[Download Code](https://github.com/abhisheksaini4/CustomRatingBarExample)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) and name it CustomRatingBarExample

Select File -> New -> New Project -> then Fill the forms and click "Finish" button.

**Step 2:** Open res -> layout -> activity\_main.xml (or) main.xml and add following code

In this step we open an xml file and add the code for displaying a custom rating bar with five number of stars and “ 2.5 ” value for default rating and one submit button.

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#fff"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<RatingBar

android:id="@+id/simpleRatingBar"

style="@style/customRatingBar"

android:layout\_width="wrap\_content"

android:layout\_height="60dp"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="60dp"

android:background="#0f0"

android:rating="2.5" />

<Button

android:id="@+id/submitButton"

android:layout\_width="200dp"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:background="#f40"

android:padding="10dp"

android:text="Submit"

android:textColor="#fff"

android:textSize="20sp"

android:textStyle="bold" />

</RelativeLayout>

**Step 3:** Open src -> package -> **MainActivity.java**

In this step we open MainActivity where we add the code to initiate the RatingBar & button and then perform click event on button and display the total number of stars and rating by using a Toast.

package example.gb.customratingbarexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.RatingBar;

import android.widget.Button;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// initiate rating bar and a button

final RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.simpleRatingBar);

Button submitButton = (Button) findViewById(R.id.submitButton);

// perform click event on button

submitButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// get values and then displayed in a toast

String totalStars = "Total Stars:: " + simpleRatingBar.getNumStars();

String rating = "Rating :: " + simpleRatingBar.getRating();

Toast.makeText(getApplicationContext(), totalStars + "\n" + rating, Toast.LENGTH\_LONG).show();

}

});

}

}

**Step 4:** Open res ->values ->**styles.xml**

In this step we add the code for displaying custom stars for rating. To do that in this styles.xml file we set an custom xml file from drawable and set the height and width for that item.

<resources>

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

</style>

<style name="customRatingBar" parent="@android:style/Widget.RatingBar">

<item name="android:progressDrawable">@drawable/customratingstars</item>

<item name="android:minHeight">20dip</item>

<item name="android:maxHeight">20dip</item>

</style>

</resources>

**Step 5:**Create a new drawable XML file –>**customratingstars.xml**

In this step we create a new drawable XML file, in which we set the icons for filled and empty stars. As shown in below code snippet empty and filled are two different icons set from drawable.

<?xml version="1.0" encoding="utf-8"?>

<layer-list xmlns:android="http://schemas.android.com/apk/res/android">

<item android:id="@android:id/background"

android:drawable="@drawable/empty" />

<item android:id="@android:id/secondaryProgress"

android:drawable="@drawable/empty" />

<item android:id="@android:id/progress"

android:drawable="@drawable/filled" />

</layer-list>

**Output:**

Now start the AVD in Emulator and run the App. You will see custom star rating option on screen.

## WebView Tutorial With Example In Android Studio

In Android, [WebView](https://abhiandroid.com/ui/webview/) is a view used to display the web pages in application. This class is the basis upon which you can roll your own web browser or simply use it to display some online content within your Activity. We can also specify [HTML](https://abhiandroid.com/ui/html/) string and can show it inside our application using a [WebView](https://abhiandroid.com/ui/webview/). Basically, [WebView](https://abhiandroid.com/ui/webview/) turns application into a web application.

In order to add [Web View](https://abhiandroid.com/ui/webview/) in your application, you have to add **<WebView>**element to your [XML](https://abhiandroid.com/ui/xml/)( layout ) file or you can also add it in [java](https://abhiandroid.com/java/) class.

<WebView

android:id="@+id/simpleWebView"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" />

#### ****Internet Permission Required For Webview:****

**Important Note:** In order for Activity to access the Internet and load the web pages in a WebView, we must add the internet permissions to our Android Manifest file (Manifest.[xml](https://abhiandroid.com/ui/xml/)).

Below code define the internet permission in our manifest file to access the internet in our application.

<!--Add this before application tag in AndroidManifest.xml-->

<uses-permission android:name="android.permission.INTERNET" />



#### ****Methods of WebView In Android:****

Let’s discuss some common methods of a Webview which are used to configure a [web view](https://abhiandroid.com/ui/webview/) in our application.

**loadUrl() – Load a web page in our WebView**

**loadUrl(String url)**

This function is used to load a web page in a [web view](https://abhiandroid.com/ui/webview/) of our application. In this method we specify the url of the web page that should be loaded in a web view.

Below we load a url: [https://shivatutorials.com/ui/](https://abhiandroid.com/ui/) in our application.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a web view

WebView simpleWebView=(WebView) findViewById(R.id.simpleWebView);

// specify the url of the web page in loadUrl function

simpleWebView.loadUrl("https://shivatutorials.com/ui/");

**2. loadData() – Load Static Html Data on WebView**

**loadData(String data, String mimeType, String encoding)**

This method is used to load the static [HTML](https://abhiandroid.com/ui/html/) string in a web view. loadData() function takes [html](https://abhiandroid.com/ui/html/) string data, mime-type and encoding param as three parameters.

Below we load the static Html string data in our application of a web view.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a web view

WebView webView = (WebView) findViewById(R.id.simpleWebView);

// static html string data

String customHtml = "<html><body><h1>Hello, Shivatutorials</h1>" +

"<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>" +

"<p>This is a sample paragraph of static HTML In Web view</p>" +

"</body></html>";

// load static html data on a web view

webView.loadData(customHtml, "text/html", "UTF-8");

**3. Load Remote URL on WebView using WebViewClient:**

WebViewClient help us to monitor event in a WebView. You have to Override the shouldOverrideUrlLoading() method. This method allow us to perform our own action when a particular url is selected. Once you are ready with the WebViewClient, you can set the WebViewClient in your WebView using the setWebViewClient() method.

Below we load a url by using web view client in a WebView.

/\*Add in Oncreate() funtion after setContentView()\*/

// initiate a web view

simpleWebView = (WebView) findViewById(R.id.simpleWebView);

// set web view client

simpleWebView.setWebViewClient(new MyWebViewClient());

// string url which you have to load into a web view

String url = "https://shivatutorials.com/ui/";

simpleWebView.getSettings().setJavaScriptEnabled(true);

simpleWebView.loadUrl(url); // load the url on the web view

}

// custom web view client class who extends WebViewClient

private class MyWebViewClient extends WebViewClient {

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url); // load the url

return true;

}

**4. canGoBack() – Move to one page back if a back history exist**

This method is used to specify whether the web view has a back history item or not. This method returns a Boolean value either true or false. If it returns true then goBack() method is used to move one page back.

Below we check whether a web view has back history or not.

// initiate a web view

WebView simpleWebView=(WebView)findViewById(R.id.***simpleWebView***);

// checks whether a web view has a back history item or not

Boolean canGoBack=simpleWebView.canGoBack();

**5. canGoForward() – Move one page forward if forward history exist**

This method is used to specify whether the web view has a forword history item or not. This method returns a Boolean value either true or false. If it returns true then goForword() method is used to move one page forword.

Below we check whether a web view has forward history or not.

// initiate a web view

WebView simpleWebView=(WebView)findViewById(R.id.simpleWebView);

// checks whether a web view has a forward history item or not

Boolean canGoForword=simpleWebView.canGoForward() ;

**6. clearHistory() – clear the WebView history**

This method is used to clear the web view forward and backward history.

Below we clear the forword and backword history of a WebView.

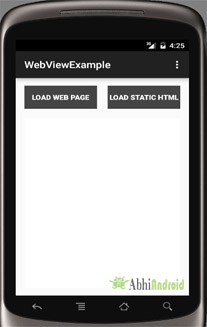
WebView simpleWebView=(WebView)findViewById(R.id.***simpleWebView***); // initiate a web view

simpleWebView.clearHistory(); // clear the forward and backward history

#### ****WebView Example In Android Studio:****

Here in this WebView example we show the use of web view in our application. To do that we display two buttons one is for displaying a web page and other is for displaying static HTML data in a web view. Below is the final output, download code and step by step explanation:

[Download Code](https://github.com/abhisheksaini4/WebViewExample)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) and name it **WebViewExample**

Select File -> New -> New Project… then Fill the forms and click "Finish" button.

**Step 2:** Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml/) (or) main.xml and add following code :

In this step we open an XML file and add the code for displaying two buttons and a Webview in our xml file ( layout ).

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:weightSum="2">

<Button

android:id="@+id/loadWebPage"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginRight="10dp"

android:layout\_weight="1"

android:background="#444"

android:text="Load Web Page"

android:textColor="#fff"

android:textSize="14sp"

android:textStyle="bold" />

<Button

android:id="@+id/loadFromStaticHtml"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="10dp"

android:layout\_weight="1"

android:background="#444"

android:text="Load Static HTML"

android:textColor="#fff"

android:textSize="14sp"

android:textStyle="bold" />

</LinearLayout>

<WebView

android:id="@+id/simpleWebView"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:layout\_marginTop="20dp"

android:scrollbars="none" />

</LinearLayout>

**Step 3:** Open   src -> package -> MainActivity.[java](https://abhiandroid.com/java/)

In this step we open MainActivity and add the code to initiate the web view and two buttons. Out of those one [button](https://abhiandroid.com/ui/button/) is used for displaying a web page in a webview and other one is used to load a static HTML page in webview.

package example.gb.webviewexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.ButtonBarLayout;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.webkit.WebView;

import android.webkit.WebViewClient;

import android.widget.Button;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

WebView simpleWebView;

Button loadWebPage, loadFromStaticHtml;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// initiate buttons and a web view

loadFromStaticHtml = (Button) findViewById(R.id.loadFromStaticHtml);

loadFromStaticHtml.setOnClickListener(this);

loadWebPage = (Button) findViewById(R.id.loadWebPage);

loadWebPage.setOnClickListener(this);

simpleWebView = (WebView) findViewById(R.id.simpleWebView);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.loadFromStaticHtml:

// define static html text

String customHtml = "<html><body><h1>Hello, Shivatutorials</h1>" +

"<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>" +

"<p>This is a sample paragraph of static HTML In Web view</p>" +

"</body></html>";

simpleWebView.loadData(customHtml, "text/html", "UTF-8"); // load html string data in a web view

break;

case R.id.loadWebPage:

simpleWebView.setWebViewClient(new MyWebViewClient());

String url = "https://shivatutorials.com/ui/";

simpleWebView.getSettings().setJavaScriptEnabled(true);

simpleWebView.loadUrl(url); // load a web page in a web view

break;

}

}

private class MyWebViewClient extends WebViewClient {

@Override

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url);

return true;

}

}

}

**Step 4:** Open **manifests -> AndroidManifest.xml**

In this step we open Manifest file and define the internet permission for our app.

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="example.gb.webviewexample">

<!-- define internet permission for our app -->

<uses-permission android:name="android.permission.INTERNET" />

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme">

<activity

android:name=".MainActivity"

android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

**Output:**

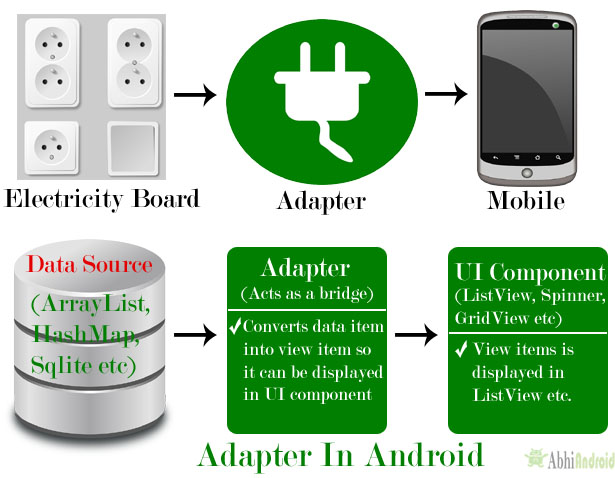
Now start the AVD in Emulator and run the App. Choose either to open webpage or static HTML in WebView by clicking on [Button](https://abhiandroid.com/ui/button/). We open static HTML.

Adapter Tutorial With Example In Android Studio

n Android, [Adapter](https://abhiandroid.com/ui/adapter/) is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an [Adapter](https://abhiandroid.com/ui/adapter/) view then view can takes the data from the [adapter](https://abhiandroid.com/ui/adapter/) view and shows the data on different views like as [ListView](https://abhiandroid.com/ui/listview/), [GridView](https://abhiandroid.com/ui/gridview/), [Spinner](https://abhiandroid.com/ui/spinner/) etc. For more customization in Views we uses the base adapter or custom adapters.

To fill data in a list or a grid we need to implement Adapter. Adapters acts like a bridge between UI component and data source. Here data source is the source from where we get the data and UI components are list or grid items in which we want to display that data.

**Below is a conceptual diagram of Adapter:**



#### ****Adapters In Android:****

There are the some commonly used Adapter in Android used to fill the data in the UI components.

1. BaseAdapter – It is parent adapter for all other adapters
2. ArrayAdapter – It is used whenever we have a list of single items which is backed by an array
3. Custom ArrayAdapter – It is used whenever we need to display a custom list
4. SimpleAdapter – It is an easy adapter to map static data to views defined in your [XML](https://abhiandroid.com/ui/xml/) file
5. Custom SimpleAdapter – It is used whenever we need to display a customized list and needed to access the child items of the list or grid

## Spinner Tutorial With Examples In Android Studio

In Android, [Spinner](https://abhiandroid.com/ui/spinner/) provides a quick way to select one value from a set of values. Android spinners are nothing but the drop down-list seen in other programming languages. In a default state, a [spinner](https://abhiandroid.com/ui/spinner/) shows its currently selected value. It provides a easy way to select a value from a list of values.

n Simple Words we can say that a [spinner](https://abhiandroid.com/ui/spinner/) is like a combo box of AWT or swing where we can select a particular item from a list of items. Spinner is a sub class of AsbSpinner class.

**Important Note:**Spinner is associated with [Adapter](https://abhiandroid.com/ui/adapter/) view so to fill the data in spinner we need to use one of the [Adapter](https://abhiandroid.com/ui/adapter/) class.

**Here is the XML basic code for Spinner:**

<Spinner

android:id="@+id/simpleSpinner "

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content" />

**Important Note:** To fill the data in a spinner we need to implement an [adapter](https://abhiandroid.com/ui/adapter/) class. A spinner is mainly used to display only text field so we can implement Array Adapter for that. We can also use [Base Adapter](https://abhiandroid.com/ui/baseadapter-tutorial-example.html) and other custom adapters to display a spinner with more customize list. Suppose if we need to display a [textview](https://abhiandroid.com/ui/textview/) and a [imageview](https://abhiandroid.com/ui/imageview/) in spinner item list then [array adapter](https://abhiandroid.com/ui/arrayadapter-tutorial-example.html) is not enough for that. Here we have to implement custom adapter in our class. Below image of Spinner and Custom Spinner will make it more clear.



**ArraryAdapter:**

An adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to adapter view then view can takes the data from the adapter view and shows the data on different views like as [list view](https://abhiandroid.com/ui/listview/), [grid view](https://abhiandroid.com/ui/gridview/), spinner. Whenever you have a list of single items which is backed by an array, you can use Array Adapter.

**Here is code of ArrayAdapter in Android:**

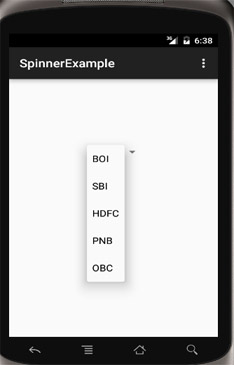
ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)

For more details read [ArrayAdapter Tutorial](https://abhiandroid.com/ui/arrayadapter-tutorial-example.html) as here we will use it in the below example to explain how Spinner is created in Android.

#### ****Example of Spinner In Android Studio:****

**Example 1:** Below is the example in which we display a list of bank names in a spinner and whenever you select an item the value will be displayed using toast on Mobile screen. Below is the final output and code:

[Download Code](https://github.com/abhisheksaini4/SpinnerExamples)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) in [Android Studio](https://abhiandroid.com/androidstudio/)and name it SpinnerExample.

Select File -> New -> New Project ->. Fill the forms and click "Finish" button.

**Step 2:** Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml/) (or) main.[xml](https://abhiandroid.com/ui/xml/) and add following code. Here we will create a Spinner inside [Relative Layout](https://abhiandroid.com/ui/relative-layout/).

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<Spinner

android:id="@+id/simpleSpinner"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="100dp" />

</RelativeLayout>

**Step 3:** Now open app-> [java](https://abhiandroid.com/java/) -> package -> MainActivity.[java](https://abhiandroid.com/java/) and add the following code. Here we will use ArrayAdapter to fill the data in Spinner. Also we are using Toast to display when the item in Spinner is selected.

package example.abhiandriod.spinnerexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Spinner;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener{

String[] bankNames={"BOI","SBI","HDFC","PNB","OBC"};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

*//Getting the instance of Spinner and applying OnItemSelectedListener on it*

Spinner spin = (Spinner) findViewById(R.id.*simpleSpinner*);

spin.setOnItemSelectedListener(this);

*//Creating the ArrayAdapter instance having the bank name list*

ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.*simple\_spinner\_item*,bankNames);

aa.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);

*//Setting the ArrayAdapter data on the Spinner*

spin.setAdapter(aa);

}

*//Performing action onItemSelected and onNothing selected*

@Override

public void onItemSelected(AdapterView<?> arg0, View arg1, int position,long id) {

Toast.*makeText*(getApplicationContext(), bankNames[position], Toast.*LENGTH\_LONG*).show();

}

@Override

public void onNothingSelected(AdapterView<?> arg0) {

*// TODO Auto-generated method stub*

}

}

#### ****Custom Spinner:****

Custom Spinner is used to display a spinner item with image, text etc (i.e. creating more custom list item). It is achieved in Android using custom adapter like base adapter. For more details read [Custom Spinner Tutorial](https://abhiandroid.com/ui/custom-spinner-examples.html).



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) in [Android Studio](https://abhiandroid.com/androidstudio/)and name it CustomSpinnerExample.

Select File -> New -> New Project. Fill the requirements and click "Finish" button.

**Step 2:** Open res -> layout -> [xml](https://abhiandroid.com/ui/xml/) (or) activity\_main.[xml](https://abhiandroid.com/ui/xml/) and add following code. Here we are creating Spinner inside [Relative Layout](https://abhiandroid.com/ui/relative-layout/).

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<Spinner

android:id="@+id/simpleSpinner"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="50dp" />

</RelativeLayout>

**Step 3:** Create a new layout activity in res-> layout and name it custom\_spinner\_items.xml. Add following code. Here we are defining basic layout for custom items that will be displayed inside Spinner.

<?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="wrap\_content"

android:orientation="horizontal">

<ImageView

android:id="@+id/imageView"

android:layout\_width="50dp"

android:layout\_height="50dp"

android:padding="5dp"

android:src="@drawable/ic\_launcher" /><!--Make sure image is present in Drawable folder-->

<TextView

android:id="@+id/textView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:padding="@dimen/activity\_horizontal\_margin"

android:text="Demo"

android:textColor="#000" />

</LinearLayout>

**Step 4:** Open app -> [java](https://abhiandroid.com/java/) -> package -> MainActivity.[java](https://abhiandroid.com/java/) and add the following code. Explanation is included in the code itself.

package example.abhiandriod.customspinnerexample;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.Spinner;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener{

String[] countryNames={"India","China","Australia","Portugle","America","New Zealand"};

int flags[] = {R.drawable.india, R.drawable.china, R.drawable.australia, R.drawable.portugle, R.drawable.america, R.drawable.new\_zealand};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//Getting the instance of Spinner and applying OnItemSelectedListener on it

Spinner spin = (Spinner) findViewById(R.id.simpleSpinner);

spin.setOnItemSelectedListener(this);

CustomAdapter customAdapter=new CustomAdapter(getApplicationContext(),flags,countryNames);

spin.setAdapter(customAdapter);

}

//Performing action onItemSelected and onNothing selected

@Override

public void onItemSelected(AdapterView<?> arg0, View arg1, int position,long id) {

Toast.makeText(getApplicationContext(), countryNames[position], Toast.LENGTH\_LONG).show();

}

@Override

public void onNothingSelected(AdapterView<?> arg0) {

// TODO Auto-generated method stub

}

}

**Step 5:** Create a new Class app -> [java](https://abhiandroid.com/java/)-> package and new class name CustomAdapter.java and add the following code. Here we will [override the methods](https://abhiandroid.com/java/method-overriding) of BaseAdapter to fill data in Spinner.

package example.abhiandriod.customspinnerexample;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.TextView;

public class CustomAdapter extends BaseAdapter {

Context context;

int flags[];

String[] countryNames;

LayoutInflater inflter;

public CustomAdapter(Context applicationContext, int[] flags, String[] countryNames) {

this.context = applicationContext;

this.flags = flags;

this.countryNames = countryNames;

inflter = (LayoutInflater.from(applicationContext));

}

@Override

public int getCount() {

return flags.length;

}

@Override

public Object getItem(int i) {

return null;

}

@Override

public long getItemId(int i) {

return 0;

}

@Override

public View getView(int i, View view, ViewGroup viewGroup) {

view = inflter.inflate(R.layout.custom\_spinner\_items, null);

ImageView icon = (ImageView) view.findViewById(R.id.imageView);

TextView names = (TextView) view.findViewById(R.id.textView);

icon.setImageResource(flags[i]);

names.setText(countryNames[i]);

return view;

}

}

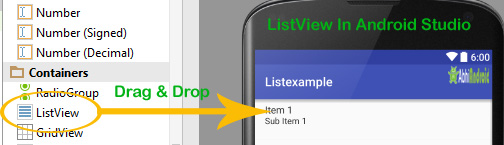
## ListView Tutorial With Example In Android Studio

List of scrollable items can be displayed in Android using [ListView](https://abhiandroid.com/ui/listview/). It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it. [ListView](https://abhiandroid.com/ui/listview/) is default scrollable so we do not need to use scroll View or anything else with [ListView](https://abhiandroid.com/ui/listview/).

ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

**Adapter:** To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an [Adapter](https://abhiandroid.com/ui/adapter/) that pulls the content from a source such as an arraylist, array or database.

**ListView in Android Studio:** Listview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it. Alternatively you can also [XML](https://abhiandroid.com/ui/xml/) code to create it.



**Here is Android ListView XML Code:**

<ListView xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/simpleListView"

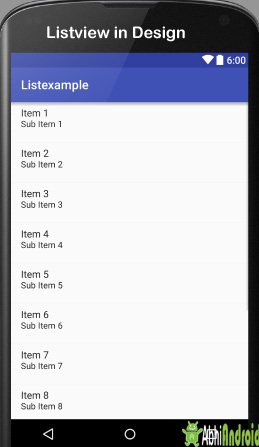
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

tools:context="abhiandroid.com.listexample.MainActivity">

</ListView>

**Listview look in Design:**



**Table Of Contents**[[hide](https://abhiandroid.com/ui/listview)]

* [1 Attributes of ListView:](https://abhiandroid.com/ui/listview#Attributes_of_ListView)
* [2 Adapters Use in ListView:](https://abhiandroid.com/ui/listview#Adapters_Use_in_ListView)

#### ****Attributes of ListView:****

Lets see some different attributes of ListView which will be used while designing a custom list:

**1. id:** id is used to uniquely identify a ListView.

Below is the id attribute’s example code with explanation included.

<!-- Id of a list view uniquely identify it-->

<ListView

android:id="@+id/simpleListView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

/>

**2. divider:**This is a drawable or color to draw between different list items.

Below is the divider example code with explanation included, where we draw red color divider between different views.

<!--Divider code in ListView-->

<ListView

android:id="@+id/simpleListView"

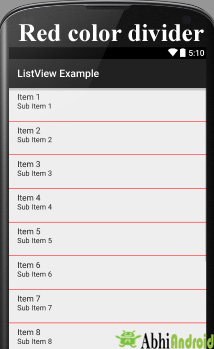
android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:divider="#f00"

android:dividerHeight="1dp"

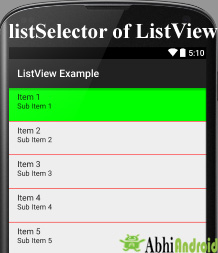
/>



**3. dividerHeight:** This specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).

In above example of divider we also set the divider height 1dp between the list items. The height should be in dp,sp or px.

**4. listSelector:**listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your design.



Below is listSelector example code with explanation includes, where list selector color is green, when you select any list item then that item’s background color is green .

**<!-- List Selector Code in ListView -->**

<ListView

android:id="@+id/simpleListView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:divider="#f00"

android:dividerHeight="1dp"

android:listSelector="#0f0"/> <!--list selector in green color-->

#### ****Adapters Use in ListView:****

An [adapter](https://abhiandroid.com/ui/adapter/) is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to [adapter](https://abhiandroid.com/ui/adapter/) view then view can takes the data from the adapter view and shows the data on different views like as [list view](https://abhiandroid.com/ui/listview/), [grid view](https://abhiandroid.com/ui/gridview/), [spinner](https://abhiandroid.com/ui/spinner/) etc.

ListView is a subclass of AdapterView and it can be populated by binding  to an Adapter, which retrieves the data from an external source and creates a View that represents each data entry.

**In android commonly used adapters are:**

1. Array Adapter
2. Base Adapter

Now we explain these two adapter in detail:

**1.Array Adapter:**

Whenever you have a list of single items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.

**Important Note:** By default, ArrayAdapter expects a Layout with a single [TextView](https://abhiandroid.com/ui/textview/), If you want to use more complex views means more customization in list items, please avoid ArrayAdapter and use custom adapters.

Below is Array Adapter code:

ArrayAdapter adapter = new ArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);

**Example of list view using Array Adapter**:

In this example, we display a list of countries by using simple array adapter. Below is the final output we will create:

**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) Listexample and activity Main Activity. Here we will create a ListView in LinearLayout. Below is the code of activity\_main.[xml](https://abhiandroid.com/ui/xml/) or content\_main.[xml](https://abhiandroid.com/ui/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">

<ListView

android:id="@+id/simpleListView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:divider="@color/material\_blue\_grey\_800"

android:dividerHeight="1dp" />

</LinearLayout>

**Step 2:** Create a new activity name Listview and below is the code of activity\_listview.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">

<TextView

android:id="@+id/textView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:padding="@dimen/activity\_horizontal\_margin"

android:textColor="@color/black" />

</LinearLayout>

**Step 3:** Now in this final step we will use ArrayAdapter to display the country names in UI. **Below is the code of MainActivity.java**

package abhiandroid.com.listexample;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.widget.ArrayAdapter;import android.widget.ListView;

public class MainActivity extends Activity

{

// Array of strings...

ListView simpleList;

String countryList[] = {"India", "China", "australia", "Portugle", "America", "NewZealand"};

@Override protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

simpleList = (ListView)findViewById(R.id.simpleListView);

ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, R.layout.activity\_listview, R.id.textView, countryList);

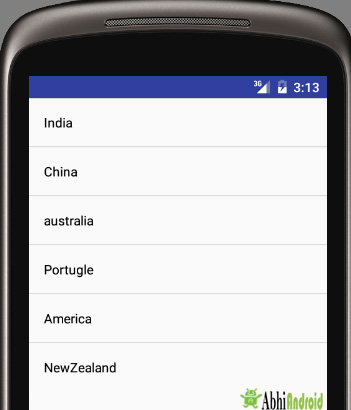
simpleList.setAdapter(arrayAdapter);

}

}

**Output Screen:**

Now run the App in Emulator. You will see the below output screen where list of country names will be printed:



**2.Base Adapter:**

BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView. Whenever you need a customized list you create your own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item.  ArrayAdapter is also an implementation of BaseAdapter.

**Example of list view using Custom adapter(Base adapter):**

In this example we display a list of countries with flags. For this, we have to use custom adapter as shown in example:



**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) Listebasexample and activity Main Activity. Here we will create a ListView in LinearLayout. **Below is the code of activity\_main.xml or content\_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">

<ListView

android:id="@+id/simpleListView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:divider="@color/material\_blue\_grey\_800"

android:dividerHeight="1dp"

android:footerDividersEnabled="false" />

</LinearLayout>

**Step 2:** Create a new activity name Listview and below is the code of activity\_listview.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="horizontal">

<ImageView

android:id="@+id/icon"

android:layout\_width="50dp"

android:layout\_height="50dp"

android:src="@drawable/ic\_launcher" />

<TextView

android:id="@+id/textView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:padding="@dimen/activity\_horizontal\_margin"

android:textColor="@color/black" />

</LinearLayout>

**Step 3:** In third step we will use custom adapter to display the country names in UI by coding MainActivity.[java](https://abhiandroid.com/java/). **Below is the code of MainActivity.java**

**Important Note:** Make sure flag images are stored in drawable folder present inside res folder with correct naming.

package com.abhiandroid.listbaseexample;

import android.app.Activity;

import android.os.Bundle;

import android.widget.ListView;

public class MainActivity extends Activity {

ListView simpleList;

String countryList[] = {"India", "China", "australia", "Portugle", "America", "NewZealand"};

int flags[] = {R.drawable.india, R.drawable.china, R.drawable.australia, R.drawable.portugle, R.drawable.america, R.drawable.new\_zealand};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

simpleList = (ListView) findViewById(R.id.simpleListView);

CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(), countryList, flags);

simpleList.setAdapter(customAdapter);

}

}

**Step 4:**Now create another class Custom Adapter which will extend BaseAdapter. Below is the code of CustomAdapter.[java](https://abhiandroid.com/java/)

package com.abhiandroid.listbaseexample;

import android.content.Context;

import android.media.Image;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.TextView;

import java.util.zip.Inflater;

public class CustomAdapter extends BaseAdapter {

Context context;

String countryList[];

int flags[];

LayoutInflater inflter;

public CustomAdapter(Context applicationContext, String[] countryList, int[] flags) {

this.context = context;

this.countryList = countryList;

this.flags = flags;

inflter = (LayoutInflater.*from*(applicationContext));

}

@Override

public int getCount() {

return countryList.length;

}

@Override

public Object getItem(int i) {

return null;

}

@Override

public long getItemId(int i) {

return 0;

}

@Override

public View getView(int i, View view, ViewGroup viewGroup) {

view = inflter.inflate(R.layout.*activity\_listview*, null);

TextView country = (TextView)           view.findViewById(R.id.*textView*);

ImageView icon = (ImageView) view.findViewById(R.id.*icon*);

country.setText(countryList[i]);

icon.setImageResource(flags[i]);

return view;

}

**Output:**

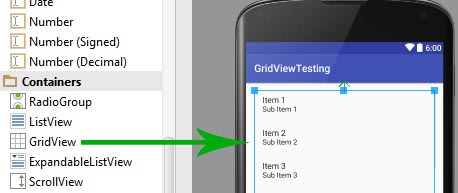
## GridView Tutorial With Examples In Android

In android [GridView](https://abhiandroid.com/ui/gridview/) is a view group that display items in two dimensional scrolling grid (rows and columns), the grid items are not necessarily predetermined but they are automatically inserted to the layout using a ListAdapter. Users can then select any grid item by clicking on it. [GridView](https://abhiandroid.com/ui/gridview/) is default scrollable so we don’t need to use [ScrollView](https://abhiandroid.com/ui/scrollview/) or anything else with [GridView](https://abhiandroid.com/ui/gridview/).

GridView is widely used in android applications. An example of GridView is your default [Gallery](https://abhiandroid.com/ui/gallery/), where you have number of images displayed using grid.

**Adapter Is Used To Fill Data In Gridview:** To fill the data in a GridView we simply use [adapter](https://abhiandroid.com/ui/adapter/) and grid items are automatically inserted to a GridView using an [Adapter](https://abhiandroid.com/ui/adapter/) which pulls the content from a source such as an arraylist, array or database. You can read full [Adapter tutorial here](https://abhiandroid.com/ui/adapter).

**GridView in Android Studio:** Gridview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it. Alternatively you can also [XML](https://abhiandroid.com/ui/xml/) code to create it.



**Basic GridView code in XML:**

<GridView

android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:numColumns="3"/>

#### ****Attributes of GridView:****

Lets see different attributes of GridView which will be used while designing a custom [grid view](https://abhiandroid.com/ui/gridview/):

**1.id:** id is used to uniquely identify a GridView.

Below is the id attribute’s example code with explanation included in which we don’t specify the number of columns in a row that’s why the GridView behaves like a [ListView](https://abhiandroid.com/ui/listview/).

Below is the id attribute example code for Gridview:

<GridView

android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

/>

**2.numColumns:** numColumn define how many columns to show. It  may be a integer value, such as “5” or auto\_fit.

**auto\_fit**  is used to display as many columns as possible to fill the available space on the screen.

**Important Note:**If we don’t specify numColumn property in GridView it behaves like a [ListView](https://abhiandroid.com/ui/listview/) with singleChoice.

Below is the numColumns example code where we define 4 columns to show in the screen.

**<!-- numColumns example code -->**

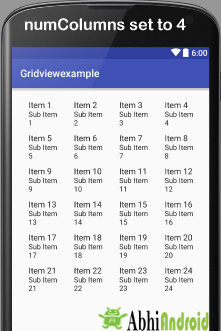
<GridView

android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:numColumns="4"/> <!-- numColumns set to 4-->



**3. horizontalSpacing:** horizontalSpacing property is used to define the default horizontal spacing between columns. This could be in pixel(px),density pixel(dp) or scale independent pixel(sp).

Below is the horizontalSpacing example code with explanation included where horizontal spacing between grid items is 50 dp.

<!--Horizontal space example code in grid view-->>

<GridView

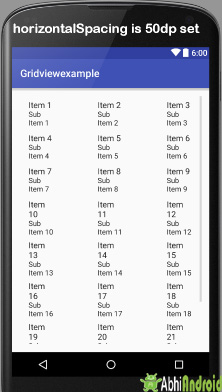
android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:numColumns="3"

android:horizontalSpacing="50dp"/><!--50dp horizontal space between grid items-->



**4.verticalSpacing:** verticalSpacing property used to define the default vertical spacing between rows. This should be in px, dp or sp.

Below is the verticalSpacing example code with explanation included, where vertical spacing between grid items is 50dp.

**<!-- Vertical space between grid items code -->**

<GridView

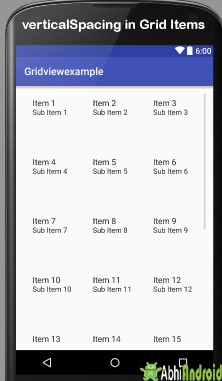
android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:numColumns="3"

android:verticalSpacing="50dp"/><!--50dp vertical space set between grid items-->



**5.columnWidth:** columnWidth property specifies the fixed width of each column. This could be in px, dp or sp.

Below is the columnWidth example code. Here column width is 80dp and selected item’s background color is green which shows the actual width of a grid item.

**Important Note:** In the below code we also used listSelector property which define color for selected item. Also to see the output of columnWidth using listSelector we need to use [Adapter](https://abhiandroid.com/ui/adapter/) which is our next topic. The below code is not sufficient to show you the output.

**<!--columnWidth in Grid view code-->**

<GridView

android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:numColumns="3"

android:columnWidth="80dp"

android:listSelector="#0f0"/><!--define green color for selected item-->



#### ****GridView Example Using Different Adapters In Android Studio:****

An adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and sends the data to adapter view, then view can takes the data from the adapter view and shows the data on different views like as [list view](https://abhiandroid.com/ui/listview/), [grid view](https://abhiandroid.com/ui/gridview/), [spinner](https://abhiandroid.com/ui/spinner/) etc.

GridView and ListView both are subclasses of AdapterView and it can be populated by binding  to an Adapter, which retrieves the data from an external source and creates a View that represents each data entry. In android commonly used adapters which fill data in GridVieware:

1. Array Adapter

2. Base Adapter

3. Custom Array Adapter

Now we explain these adapters in detail:

**1. Avoid Array Adapter To Fill Data In GridView:**

Whenever you have a list of single items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.

By default, ArrayAdapter expects a Layout with a single [TextView](https://abhiandroid.com/ui/textview/), If you want to use more complex views means more customization in grid items, please avoid ArrayAdapter and use custom adapters.

ArrayAdapter adapter = new ArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);

#### ****2. GridView Using Base Adapter In Android:****

Base Adapter is a common base class of a general implementation of an Adapter that can be used in GridView. Whenever you need a customized [grid view](https://abhiandroid.com/ui/gridview/) you create your own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying custom grid items. ArrayAdapter is also an implementation of BaseAdapter. You can read [BaseAdapter tutorial here](https://abhiandroid.com/ui/baseadapter-tutorial-example.html).

**Example of GridView using Base Adapter in Android Studio:** Below is the example of GridView in Android, in which we show the Android logo’s in the form of Grids. In this example firstly we create an int type array for logo images and then call the Adapter to set the data in the GridView. In this we create a CustomAdapter by extending BaseAdapter in it. At Last we implement setOnItemClickListener event on GridView and on click of any item we send that item to another Activity and show the logo image in full size.

Below you can download code, see final output and step by step explanation of the example.

[Download Code](https://github.com/abhisheksaini4/GridViewExample)[**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)



**Step 1:** Create a new Android project in [Android Studio](https://abhiandroid.com/androidstudio/)and fill all the required details. In our case we have named GridViewExample and package com.example.gourav.GridViewExample

**Step 2:** Open activity\_main.[xml](https://abhiandroid.com/ui/xml/) and paste the below code. In this we have created a Grid view insideLinear Layout and also set number of columns to 3.

<?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">

<!--

GridView with 3 value for numColumns attribute

-->

<GridView

android:id="@+id/simpleGridView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:footerDividersEnabled="false"

android:padding="1dp"

android:numColumns="3" />

</LinearLayout>

**Step 3:** : Create a new [XML](https://abhiandroid.com/ui/xml/) file and paste the below code. We have named activity\_gridview.xml.  
In this step we create a new XML file and add a [ImageView](https://abhiandroid.com/ui/imageview/) in it. This file is used in CustomAdapter to set the logo images

<?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="wrap\_content"

android:padding="1dp"

android:orientation="vertical">

<ImageView

android:id="@+id/icon"

android:layout\_width="match\_parent"

android:layout\_height="120dp"

android:scaleType="fitXY"

android:layout\_gravity="center\_horizontal"

android:src="@drawable/logo1" />

</LinearLayout>

**Step 4:**Now open drawable folder and save small size png images of different logo’s and name them like logo1,logo2 and etc.

**Step 5:** Now open MainActivity.[java](https://abhiandroid.com/java/) and paste the below code. If your package name is different, don’t copy it.  
In this step firstly we get the reference of GridView and then create a int type array for Android logo’s. After that we call the CustomAdapter and pass the array in it. At Last we implement setOnItemClickListener event on GridView and on click of any item we send that item to another Activity to show the logo image in Full Size. I have added comments in code to help you to understand the code easily so make you read the comments.

package com.example.gourav.GridViewExample;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.AdapterView;

import android.widget.GridView;

public class MainActivity extends AppCompatActivity {

GridView simpleGrid;

int logos[] = {R.drawable.logo1, R.drawable.logo2, R.drawable.logo3, R.drawable.logo4,

R.drawable.logo5, R.drawable.logo6, R.drawable.logo7, R.drawable.logo8, R.drawable.logo9,

R.drawable.logo10, R.drawable.logo11, R.drawable.logo12, R.drawable.logo13};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

simpleGrid = (GridView) findViewById(R.id.simpleGridView); // init GridView

// Create an object of CustomAdapter and set Adapter to GirdView

CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(), logos);

simpleGrid.setAdapter(customAdapter);

// implement setOnItemClickListener event on GridView

simpleGrid.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

// set an Intent to Another Activity

Intent intent = new Intent(MainActivity.this, SecondActivity.class);

intent.putExtra("image", logos[position]); // put image data in Intent

startActivity(intent); // start Intent

}

});

}

}

**Step 6:** Create a new class CustomAdapter and paste the below code.  
In this step we create a CustomAdapter class by extending BaseAdapter in it. In this step we set the logo image’s in the grid items. I have added comments in code to help you to understand the code easily so make you read the comments.

package com.example.gourav.GridViewExample;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

public class CustomAdapter extends BaseAdapter {

Context context;

int logos[];

LayoutInflater inflter;

public CustomAdapter(Context applicationContext, int[] logos) {

this.context = applicationContext;

this.logos = logos;

inflter = (LayoutInflater.from(applicationContext));

}

@Override

public int getCount() {

return logos.length;

}

@Override

public Object getItem(int i) {

return null;

}

@Override

public long getItemId(int i) {

return 0;

}

@Override

public View getView(int i, View view, ViewGroup viewGroup) {

view = inflter.inflate(R.layout.activity\_gridview, null); // inflate the layout

ImageView icon = (ImageView) view.findViewById(R.id.icon); // get the reference of ImageView

icon.setImageResource(logos[i]); // set logo images

return view;

}

}

**Step 7:** Now Create a new XML file named activity\_second and paste the below code in it.  
In this step we create an XML file for our Second Activity to display the logo image in full size.

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:background="#fff"

tools:context="com.example.gourav.GridViewExample.SecondActivity">

<ImageView

android:id="@+id/selectedImage"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:scaleType="fitXY" />

</RelativeLayout>

**Step 8:** Now Create a new Activity with name SecondActivity.class and add below code in it.

In this step we create a new Activity in which firstly we initiate the [ImageView](https://abhiandroid.com/ui/imageview/) and then get the Image from our Previous Activity by using Intent object and set the same in the [ImageView](https://abhiandroid.com/ui/imageview/).

package com.example.gourav.GridViewExample;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.widget.ImageView;

public class SecondActivity extends AppCompatActivity {

ImageView selectedImage;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_second);

selectedImage = (ImageView) findViewById(R.id.selectedImage); // init a ImageView

Intent intent = getIntent(); // get Intent which we set from Previous Activity

selectedImage.setImageResource(intent.getIntExtra("image", 0)); // get image from Intent and set it in ImageView

}

}

**Output:** Now run the App and you will see different Android images in GridView. Click on the any image and full size of it will open up.

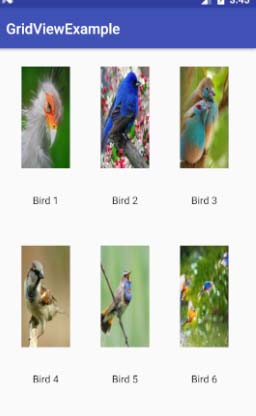
#### ****3. GridView Example Using Custom ArrayAdapter In Android Studio:****

ArrayAdapter is also an implementation of BaseAdapter so if we want more customization then we create a custom adapter and extend ArrayAdapter in that. Here we are creating GridView using [custom array adapter](https://abhiandroid.com/ui/custom-arrayadapter-tutorial-example.html).

**Example of GridView using Custom Adapter :** Example of Grid View using custom arrayadapter to show birds in the form of grids. Below is the code and final output:

Below you can download code, see final output and step by step explanation of the topic.

[Download Code](https://github.com/abhisheksaini4/GridViewExample) [**?**](https://abhiandroid.com/androidstudio/download-code-abhiandroid)

  
**Step 1:** [Create a new project](https://abhiandroid.com/androidstudio/start-create-project) and name it GridViewExample.  
**Step 2:** Now open app -> res -> layout -> activity\_main.xml (or) main.xml and add following code :

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/activity\_main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="abhiandroid.com.gridviewexample.MainActivity">

<GridView

android:id="@+id/simpleGridView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:numColumns="3"/>

</RelativeLayout>

**Step 3:** Create a new layout Activity in app -> res-> layout-> new activity and name it grid\_view\_items.xml and add following code:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/grid\_view\_items"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context="abhiandroid.com.gridviewexample.GridViewItems">

<ImageView

android:id="@+id/imageView"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:padding="5dp"

android:scaleType="fitXY"

android:src="@drawable/ic\_launcher"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true" />

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:padding="@dimen/activity\_horizontal\_margin"

android:text="Demo"

android:textColor="#000"

android:layout\_below="@+id/imageView"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="13dp" />

</RelativeLayout>

**Step 4:** Now open app -> [java](https://abhiandroid.com/java/) -> package -> MainActivity.[java](https://abhiandroid.com/java/)

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.GridView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

GridView simpleList;

ArrayList birdList=new ArrayList<>();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

simpleList = (GridView) findViewById(R.id.simpleGridView);

birdList.add(new Item("Bird 1",R.drawable.b1));

birdList.add(new Item("Bird 2",R.drawable.b2));

birdList.add(new Item("Bird 3",R.drawable.b3));

birdList.add(new Item("Bird 4",R.drawable.b4));

birdList.add(new Item("Bird 5",R.drawable.b5));

birdList.add(new Item("Bird 6",R.drawable.b6));

MyAdapter myAdapter=new MyAdapter(this,R.layout.grid\_view\_items,birdList);

simpleList.setAdapter(myAdapter);

}

}

**Step5 :** Create a new Class src -> package -> MyAdapter.java and add the following code:

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.ImageView;

import android.widget.TextView;

import java.util.ArrayList;

public class MyAdapter extends ArrayAdapter {

ArrayList birdList = new ArrayList<>();

public MyAdapter(Context context, int textViewResourceId, ArrayList objects) {

super(context, textViewResourceId, objects);

birdList = objects;

}

@Override

public int getCount() {

return super.getCount();

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

View v = convertView;

LayoutInflater inflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

v = inflater.inflate(R.layout.grid\_view\_items, null);

TextView textView = (TextView) v.findViewById(R.id.textView);

ImageView imageView = (ImageView) v.findViewById(R.id.imageView);

textView.setText(birdList.get(position).getbirdName());

imageView.setImageResource(birdList.get(position).getbirdImage());

return v;

}

}

**Step 6:**Create a new Class src -> package -> Item.java and add the below code:

public class Item {

String birdListName;

int birdListImage;

public Item(String birdName,int birdImage)

{

this.birdListImage=birdImage;

this.birdListName=birdName;

}

public String getbirdName()

{

return birdListName;

}

public int getbirdImage()

{

return birdListImage;

}

}