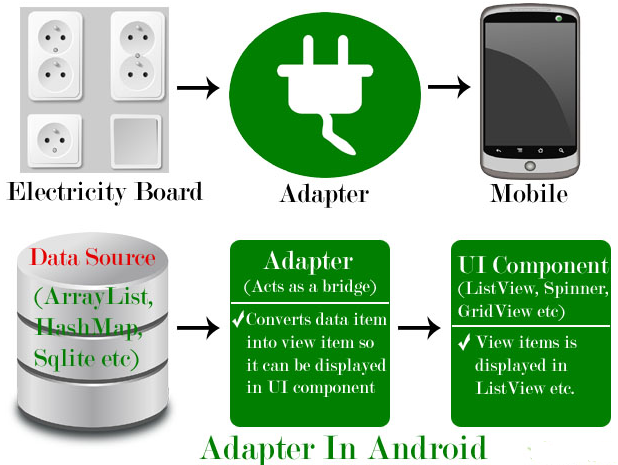
## Adapter in Android Studio

In Android, 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 an Adapter view then view can takes the data from the adapter view and shows the data on different views like as [ListView](http://abhiandroid.com/ui/listview" \t "_self), [GridView](http://abhiandroid.com/ui/gridview" \t "_self), [Spinner](http://abhiandroid.com/ui/spinner) etc. For more customization in Views we use the base adapter or custom adapters.

To fill data in a list or a grid we need to implement Adapter. Adapter’s 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:**



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

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

**1. BaseAdapter In Android:**

BaseAdapter is a common base [class](http://abhiandroid.com/java/class-objects) of a general implementation of an Adapter that can be used in ListView, GridView, Spinner etc. Whenever we need a customized list in a ListView or customized grids in a GridView we create our 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.

**Custom Adapter code which extends the BaseAdapter in that:**

public class CustomAdapter extends BaseAdapter {

@Override

public int getCount() {

return 0;

}

@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) {

return null;

}

In above code snippet we see the overrided functions of BaseAdapter which are used to set the data in a list, grid or a spinner.

**2. ArrayAdapter In Android:**

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

**Here is how android ArrayAdapter looks ::**

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

The above function are described in [ArrayAdapter tutorial with example](http://abhiandroid.com/ui/arrayadapter-tutorial-example.html).

**3. Custom ArrayAdapter In Android:**

ArrayAdapter is also an implementation of BaseAdapter, so if we want more customization then we can create a custom adapter and extend ArrayAdapter in that. Since array adapter is an implementation of BaseAdapter, so we can [override](http://abhiandroid.com/java/method-overriding) all the function’s of BaseAdapter in our custom adapter.

**Below Custom adapter class MyAdapter extends ArrayAdapter in that:**

public class MyAdapter extends ArrayAdapter {

public MyAdapter(Context context, int resource, int textViewResourceId, List objects) {

super(context, resource, textViewResourceId, objects);

}

@Override

public int getCount() {

return super.getCount();

}

@Override

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

return super.getView(position, convertView, parent);

}

}

**4. SimpleAdapter In Android:**

In Android  SimpleAdapter is an easy Adapter to map static data to views defined in an [XML](http://abhiandroid.com/ui/xml) file(layout). In Android we can specify the data backing to a list as an [ArrayList](http://abhiandroid.com/java/arraylist" \t "_self) of [Maps](http://abhiandroid.com/java/map)(i.e. [hashmap](http://abhiandroid.com/java/hashmap" \t "_self) or other). Each entry in a ArrayList is corresponding to one row of a list.

The Map contains the data for each row. Here we also specify an [XML](http://abhiandroid.com/ui/xml) file(custom list items file) that defines the views which is used to display the row, and a mapping from keys in the Map to specific views.

Whenever we have to create a custom list we need to implement custom adapter. As we discuss earlier ArrayAdapter is used when we have a list of single item’s backed by an Array. So if we need more customization in a ListView or a GridView we need to implement simple adapter.

**SimpleAdapter code in Android:**

SimpleAdapter (Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to)

**5. Custom SimpleAdapter In Android:**

Whenever we have to create a custom list we need to implement custom adapter. As we discuss earlier ArrayAdapter is used when we have a list of single item’s backed by an Array. So if we need customization in a ListView or a GridView we need to implement simple Adapter but when we need more customization in list or grid items where we have many view’s in a list item and then we have to perform any event like click or any other event to a particular view then we need to implement a custom adapter who fulfills our requirement’s and quite easy to be implemented.

BaseAdapter is the parent adapter for all other adapters so if we extends a SimpleAdapter then we can also override the base adapter’s function in that class.

**Important Note:**We can’t perform events like click and other event on child item of a list or grid but if we have some requirements to do that then we can create our own custom adapter and extends the simple adapter in that.

**Custom Adapter extends SimpleAdapter in that:**

public class CustomAdapter extends SimpleAdapter {

public CustomAdapter(Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to) {

super(context, data, resource, from, to);

}

@Override

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

return super.getView(position, convertView, parent);

}

@Override

public int getCount() {

return super.getCount();

}

}

#### ****Adapter Example In****[Android Studio](http://abhiandroid.com/androidstudio/)****:****

Below is the [Android Studio](http://abhiandroid.com/androidstudio/) example which show the use of the Adapter in Android. In this example we display a list of fruits names with images by using SimpleAdapter and whenever user click on a list item the fruit’s name displayed in a [Toast](http://abhiandroid.com/programming/custom-toast-tutorial-example.html).

Below you can download complete [Android Studio](http://abhiandroid.com/androidstudio/) code, see final output and read step by step explanation:



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

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

In this step we open an [xml](http://abhiandroid.com/ui/xml) file and add the code for displaying a ListView by using its different attributes.

<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"

tools:context=".MainActivity">

<ListView

android:id="@+id/simpleListView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:divider="#000"

android:dividerHeight="2dp"

android:listSelector="#600"/>

</RelativeLayout>

**Step 3:** Save fruit images in drawable folder with name apple, banana, litchi, mango and pineapple.

**Step 4:** Open src -> package -> MainActivity.java

In this step we add the code for initiate ListView and set the data in the list. In this firstly we create two [arrays](http://abhiandroid.com/java/arrays" \t "_self)first for fruit names and second for fruits images and then set the data in the ListView using SimpleAdapter.

package example.abhiandriod.simpleadapterexample;

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.AdapterView;

import android.widget.ListView;

import android.widget.SimpleAdapter;

import android.widget.Toast;

import java.util.ArrayList;

import java.util.HashMap;

public class MainActivity extends AppCompatActivity {

//initialize view's

ListView simpleListView;

String[] fruitsNames = {"Apple", "Banana", "Litchi", "Mango", "PineApple"};//fruit names array

int[] fruitsImages = {R.drawable.apple, R.drawable.banana, R.drawable.litchi, R.drawable.mango, R.drawable.pineapple};//fruits images

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

ArrayList<HashMap<String,String>> arrayList=new ArrayList<>();

for (int i=0;i<fruitsNames.length;i++)

{

HashMap<String,String> hashMap=new HashMap<>();//create a hashmap to store the data in key value pair

hashMap.put("name",fruitsNames[i]);

hashMap.put("image",fruitsImages[i]+"");

arrayList.add(hashMap);//add the hashmap into arrayList

}

String[] from={"name","image"};//string array

int[] to={R.id.textView,R.id.imageView};//int array of views id's

SimpleAdapter simpleAdapter=new SimpleAdapter(this,arrayList,R.layout.list\_view\_items,from,to);//Create object and set the parameters for simpleAdapter

simpleListView.setAdapter(simpleAdapter);//sets the adapter for listView

//perform listView item click event

simpleListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

Toast.makeText(getApplicationContext(),fruitsNames[i],Toast.LENGTH\_LONG).show();//show the selected image in toast according to position

}

});

}

}

**Step 5:** Create new layout-**>**rec-> layout-> list\_view\_items.[xml](http://abhiandroid.com/ui/xml) and add following code:

In this step we create a [xml](http://abhiandroid.com/ui/xml) file for displaying ListView items. In this [xml](http://abhiandroid.com/ui/xml) we add the code for displaying a [ImageView](http://abhiandroid.com/ui/imageview" \t "_self) and a [TextView](http://abhiandroid.com/ui/textview" \t "_self).

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

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="#fff">

<ImageView

android:id="@+id/imageView"

android:layout\_width="50dp"

android:layout\_height="50dp"

android:padding="5dp"

android:layout\_alignParentRight="true"

android:layout\_marginRight="10dp"

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

<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" />

</RelativeLayout>

**Output:**

Now [run the App](http://abhiandroid.com/androidstudio/run-app-avd-emulator-android-studio) and we will different fruit names listed in ListView. Here we used Simple Adapter to fill data in ListView.