OnClick() Event and OnClickListener()

# Theory

* In Android, OnClickListener is an interface provided by the View class to handle click events for UI components, such as buttons, images, or any clickable view.
* It allows you to define what actions should be performed when the user interacts with a view by clicking it.

# Features of OnClickListener:

1. Interface: View.OnClickListener is an interface that must be implemented to handle click events.
2. Single Method: It contains a single method, onClick(View v), which is called when a view is clicked.
3. Event Handling: It provides a way to programmatically set an action to be executed for a specific view's click event.

# When to Use OnClickListener?

* + When you want to handle specific logic for individual views.
  + When you need more flexibility and control over click behavior.

# Benefits of Using OnClickListener:

* + Keeps your XML layouts cleaner as no additional android:onClick attributes are required.
  + Allows separation of concerns by defining click behavior in Java/Kotlin code.
  + Provides more control and flexibility for dynamically created views.

**Basic Syntax**

view.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View v) {

// Actions to perform when the view is clicked

}

});

Explanation of the Syntax:

1. view: The UI element (e.g., Button, ImageView, etc.) that you want to make clickable.
2. setOnClickListener: A method of the View class that attaches a click listener to the view.
3. new View.OnClickListener(): Creates an anonymous inner class that implements the OnClickListener interface.
4. onClick(View v): This is the method defined in the OnClickListener interface. It gets called when the user clicks the view.

# activity\_main.xml

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

<RelativeLayout xmlns:android="[http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) xmlns:app="<http://schemas.android.com/apk/res-auto>" xmlns:tools="<http://schemas.android.com/tools>" android:layout\_width="match\_parent" android:layout\_height="match\_parent"

tools:context=".MainActivity">

<LinearLayout android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical">

<TextView android:id="@+id/tv1"

android:layout\_width="match\_parent" android:layout\_height="52dp" android:text="Enter Name:"

android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

<EditText android:id="@+id/et1"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:ems="10" android:inputType="textPersonName" />

<Button

android:id="@+id/btn1" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Submit" />

<TextView android:id="@+id/tv2"

android:layout\_width="match\_parent" android:layout\_height="39dp"

android:text="Output" android:textAppearance="@style/TextAppearance.AppCompat.Body2" />

</LinearLayout>

</RelativeLayout>

**MainActivity.java**

package com.example.androidapplicationdevelopment1;

import androidx.appcompat.app.AppCompatActivity; import android.os.Bundle;

import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.TextView; import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

TextView lbl1,lbl2; EditText txt1; Button b1; @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.*activity\_main*);

lbl1 = findViewById(R.id.*tv1*); lbl2 = findViewById(R.id.*tv2*);

txt1 = findViewById(R.id.*et1*); b1 = findViewById(R.id.*btn1*);

b1.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View view) {

String name = txt1.getText().toString(); lbl2.setText("Your name is " + name);

}

});

}

}

Output-

|  |
| --- |
| S |