

Mobile Application Development

Lesson 4

Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

Event Handling

Methods that do something in response to a click

- A method, called an **event handler**, is triggered by a specific event and does something in response to the event

Method 1: Set click handler in Java

Methods that do something in response to a click

```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String msg = "Hello Toast!";
        Toast toast = Toast.makeText(this, msg, duration);
        toast.show();
    }
});
```

Method 2: Adding Event Handler from layout.xml

- Instead of add Event handlers in Java, we can add Event Handling function for a button or any widget in the **layout.xml**.
- Or
- we can set an event handling function from the right side properties tab by setting **onClick** attribute for any widget.

Attach in XML and implement in Java

Attach handler to view in XML layout:

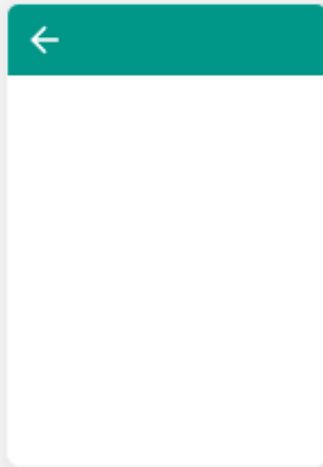
```
android:onClick="showToast"
```

Implement handler in Java activity:

```
public void showToast(View view) {  
    String msg = "Hello Toast!";  
    Toast toast = Toast.makeText(  
        this, msg, duration);  
    toast.show();  
}  
}
```

Create New Project

Configure your project



Empty Activity

Creates a new empty activity

Name

EventHandlerExample

Package name

com.example.eventhandlerexample

Save location

C:\Users\User\AndroidStudioProjects\EventHandlerExample

Language

Java

Minimum API level

API 15: Android 4.0.3 (IceCreamSandwich)

i Your app will run on approximately **100%** of devices.

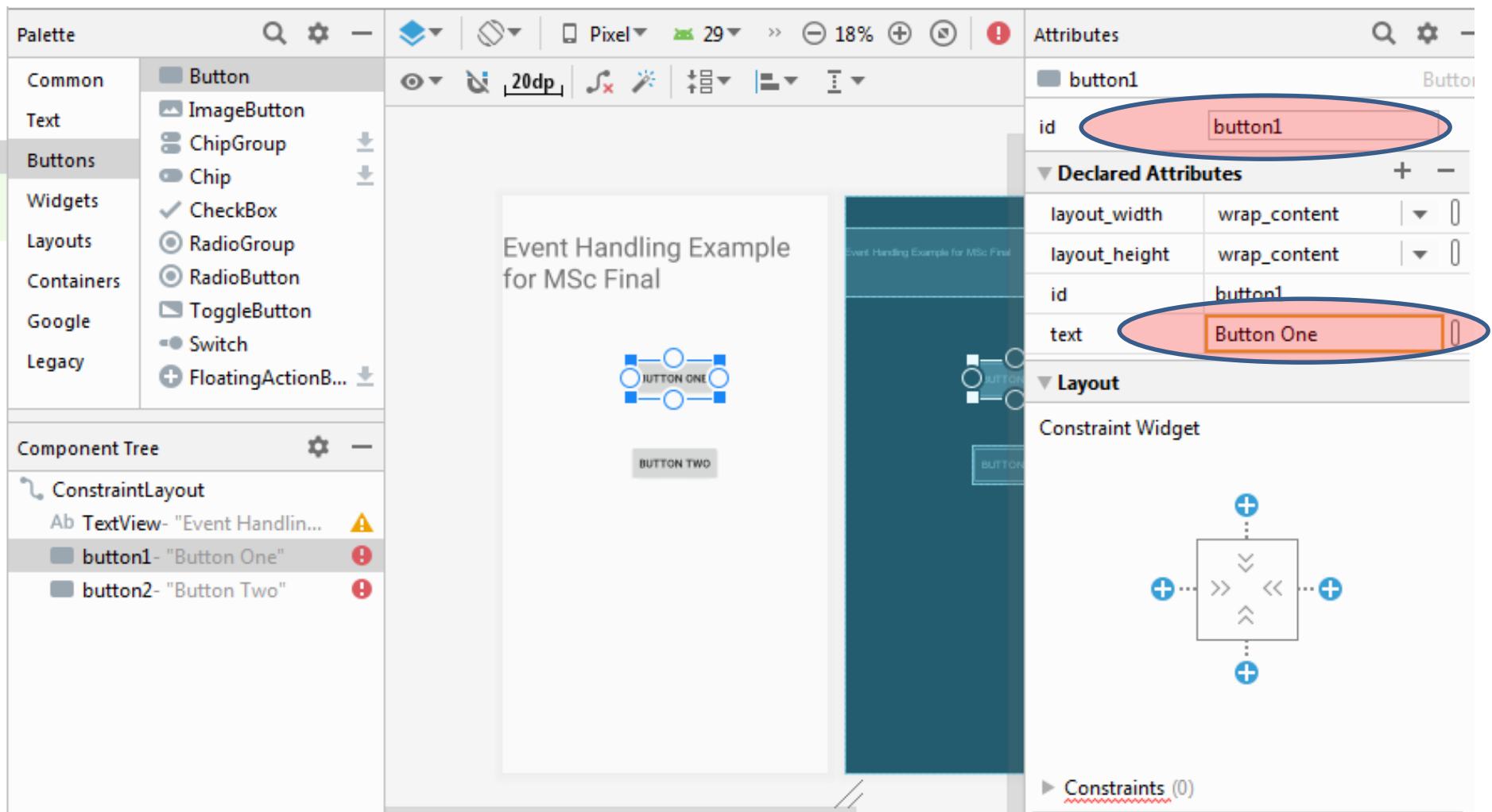
[Help me choose](#)

This project will support instant apps

Use androidx.* artifacts

Android Studio interface showing the design of a button with the following components:

- Palette:** Shows various widget categories including Buttons, Widgets, Layouts, Containers, Google, and Legacy. The **Button** widget is selected.
- Component Tree:** Lists the hierarchy: `ConstraintLayout` containing `TextView - "Event Handlin..."`, `button1 - "Button One"`, and `button2 - "Button Two"`. Red warning icons are present next to the button entries.
- Attributes:** Shows the configuration for `button1`.
 - `id`: `button1` (circled in red)
 - Declared Attributes:**
 - `layout_width`: `wrap_content`
 - `layout_height`: `wrap_content`
 - `id`: `button1`
 - `text`: `Button One` (circled in red)
 - Layout:** Shows a **Constraint Widget** diagram with a central box and four directional arrows (top, bottom, left, right) each with a plus sign, indicating wrap-content constraints.
 - Constraints:** (0)



- We have added two buttons
- Gave it ID in the properties section as button1 and button2
- Assigned Labels as
 - Button One
 - Button Two

Event Handling Example for MSc Final



BUTTON TWO

Event Handling Example for MSc Final



BUTTON ONE

visibility	<input type="text"/>	
visibility	<input type="text"/>	▼
▼ Common Attributes		
style	@android:style/Wic	▼ 0
onClick	button1Click	▼ 0
background	@android:drawable/b	0
text	Button One	0
text	<input type="text"/>	0
contentDescripti...	<input type="text"/>	0
▶ textAppearance	@android:style/Tex	▼ 0
alpha	<input type="text"/>	0
▼ All Attributes		
alpha	<input type="text"/>	0
▶ autoLink		0
autoText		0
background	@android:drawable/b	0

```
} <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="button1Click"
    android:text="Button One"
    tools:layout_editor_absoluteX="161dp"
    tools:layout_editor_absoluteY="315dp" />
}

} <Button
    android:id="@+id/button2"
    android:layout_width="wrap_co
    android:layout_height="wrap_c
    android:text="Button Two"
    tools:layout_editor_absoluteX="161dp"
    tools:layout_editor_absoluteY="315dp" />
}
```

Event
for M

- ✖ Suppress: Add tools:ignore="OnClick" attribute
- 💡 Create 'button1Click(View)' in 'MainActivity' ▶
- 🔗 Create onClick event handler ▶
- 🔗 Override Resource in Other Configuration... ▶
- 🔗 Rearrange tag attributes ▶
- 🔗 Remove attribute ▶
- 🔗 Inject language or reference ▶

In the Layout.xml for the activity, in the Button1 entry, add **android:onClick** entry and assign the name of the Event Handler function, which to be added in the java file.

Assign name and then press Alt+Enter while placing cursor on the function name. this will add event handling function in the java file.

<Button

android:id="@+id/button2"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:onClick

tools:layout_marginHorizontal

tools:keepScreenOn

```
package com.example.eventhandlerexample;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void button1Click(View view) {

    }

}
```

The function will appear in the Java file.

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void button1Click(View view) {

        TextView t1 = findViewById(R.id.textView1);
        t1.setText("You clicked Button ONE");
        Toast.makeText(context: this, text: "You Click Button one", Toast.LENGTH_LONG).show();
    }

    public void button2Click(View view) {

        TextView t2 = findViewById(R.id.textView1);
        t2.setText("You clicked Button Two");
    }
}

```

In this example, we want to change text of a Textview and also display a toast message, when button1 is clicked.

Therefore, we first access the TextView you want using findViewById(R.id.textView1). textView1 is the Id of the textview we want to use.

Using the textview object, we use it setText function to assign the text we want.

Also we create a toast message.

```
public void button2Click(View @IdRes int id
{
    TextView t1 = findViewById(R.id.)
}
```

Resource ID	Type
button1	int
button2	int
gone	int
invisible	int
packed	int
parent	int
percent	int
spread	int
spread_inside	int
textView1	int
wrap	int

inActivity > button2Click()

While you are typing to access the ID of any view, you are given suggestions to select options.

Event Handling Example to add two numbers and display results in a TextView.

- We will build an app that will add two numbers
- For this purpose we will need
 - Textview label
 - Text view for output
 - Two Text Fields: Numbers Only.
 - One button to add two numbers in the text fields.

- First we added a label TextView
- Changed the text to “Welcome to MSc Final Mobile Development” from the right side properties view.
- Increased its text size from “textAppearance”
- Changed the background color

- We also added paddings left, right, top and bottom of the main activity.

Palette

Common

- Ab TextView

Text

- Ab Plain Text
- Ab Password

Buttons

- Ab Password (Num...

Widgets

- Ab E-mail

Layouts

- Ab Phone

Containers

- Ab Postal Address
- Ab Multiline Text

Google

- Ab Time
- Ab Date

Legacy

Component Tree

- LinearLayout(vertical)
- Ab textView- "Welcome to M..."
- Ab Result- "Output"

Attributes

Ab textView

Declared Attributes

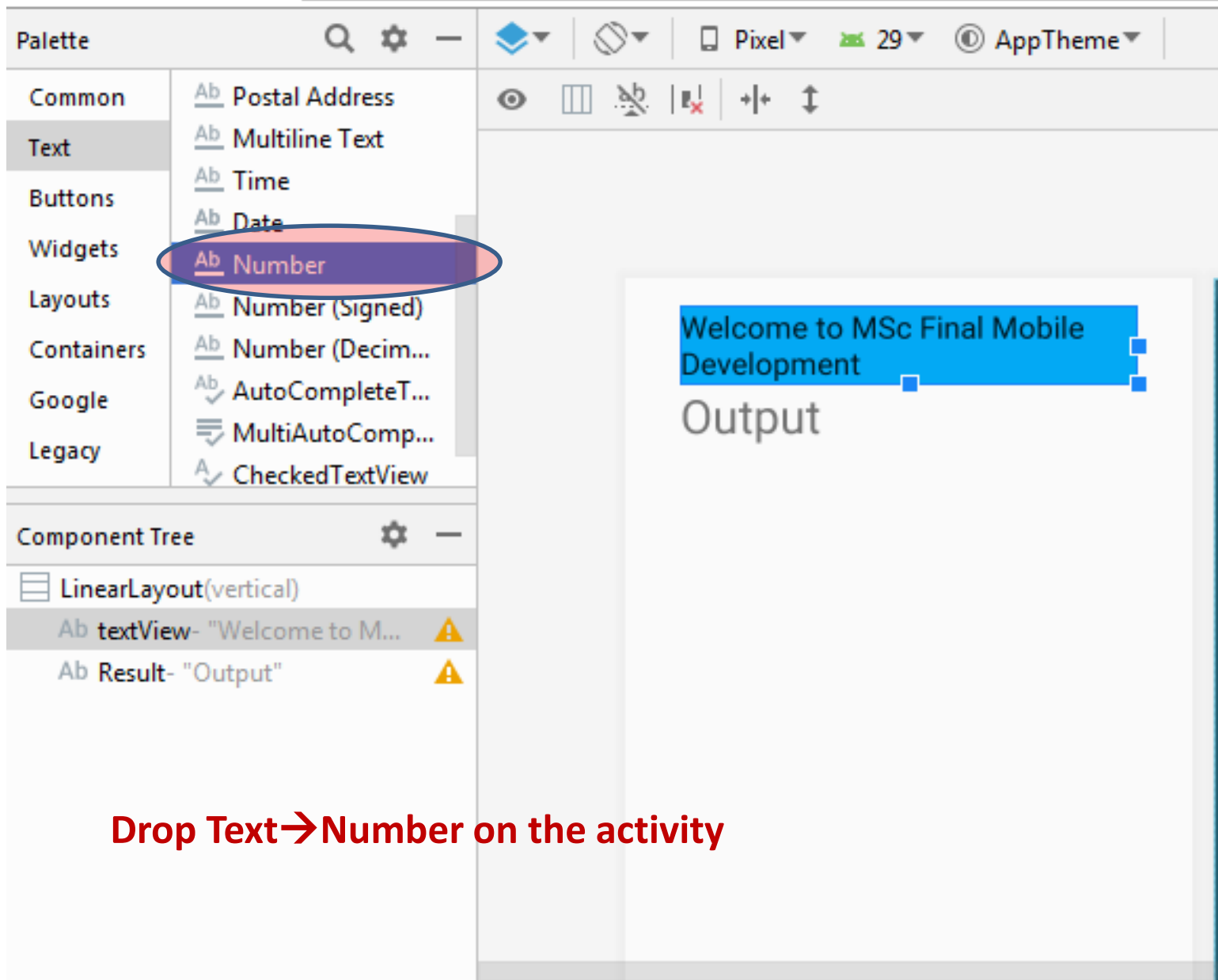
layout_width	match_parent	0
layout_height	wrap_content	0
background	#03A9F4	0
id	textView	0
text	Welcome to MSc Final M	0
textAppearance	@style/TextAppear	0

Layout

layout_width	match_parent	0
layout_height	wrap_content	0
layout_weight		0
visibility		0
visibility		0

Common Attributes

text	nal Mobile Development	0
text		0



Drop Text → Number on the activity

Pixel 29 AppTheme 25%

Attributes

idNum1 EditText

id idNum1

Declared Attributes

layout_width	match_parent	
layout_height	wrap_content	
ems	10	
hint	Enter first number	
id	idNum1	
inputType	number	

Layout

layout_width	match_parent	
layout_height	wrap_content	
layout_weight		
visibility		
visibility		

Common Attributes

inputType	number	
-----------	--------	--

Output

Welcome to MSc Final Mobile Development

Enter first number

idNum1



idNum2 EditText

id idNum2

Declared Attributes

layout_width	match_parent	0
layout_height	wrap_content	0
ems	10	0
hint	Enter Second Number	0
id	idNum2	0
inputType	number	0

Layout

layout_width	match_parent	0
layout_height	wrap_content	0
layout_weight		0
visibility		0
visibility		0

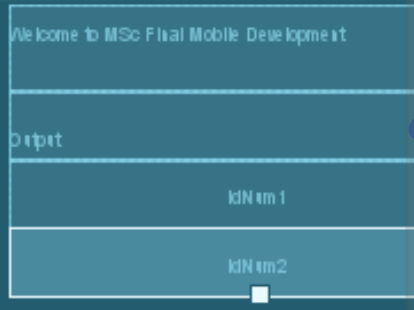
Common Attributes

Welcome to MSc Final Mobile Development

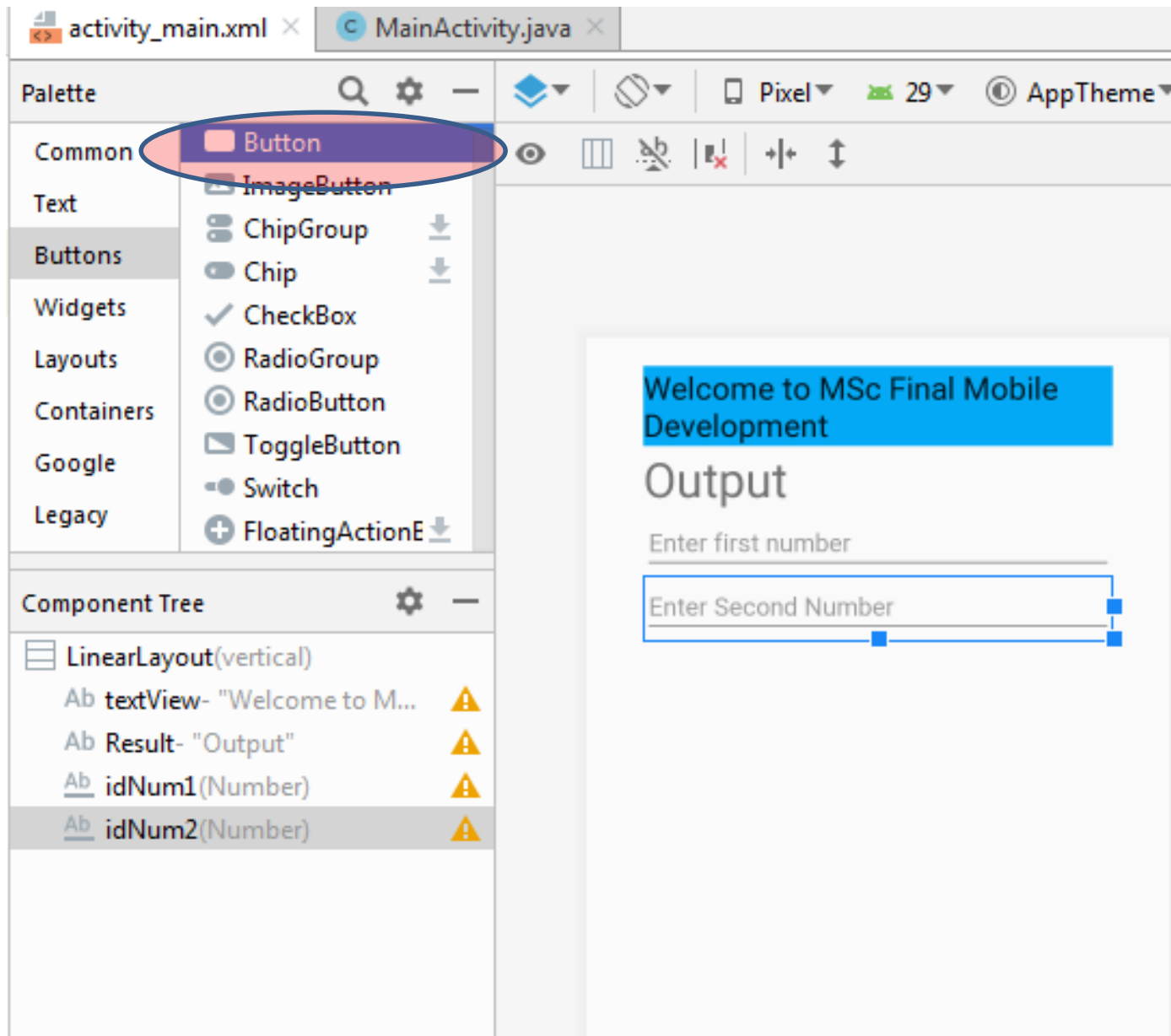
Output

Enter first number

Enter Second Number



then add Buttons -> Button



Change label of the button as **Add Values** and assign ID as **addButton**

The screenshot displays the Android Studio interface. On the left, the design view shows a mobile app layout with a blue header, an 'Output' section, two input fields, and a button labeled 'ADD VALUES'. On the right, the XML editor shows the corresponding layout code with the button's ID set to 'addButton' and its text set to 'Add Values'. The 'Attributes' panel on the far right lists the XML attributes for the 'addButton' element, with 'id' and 'text' highlighted in red ovals.

Attributes

addButton	
id	addButton
▼ Declared Attributes	
layout_width	match_parent
layout_height	wrap_content
id	addButton
text	Add Values
▼ Layout	
layout_width	match_parent
layout_height	wrap_content
layout_weight	
visibility	
visibility	
▼ Common Attributes	
style	@android:style/Wic
onClick	
background	@android:drawable/b

activity_main.xml x MainActivity.java x Pr

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical"
8     android:paddingLeft="40dp"
9     android:paddingTop="20dp"
10    android:paddingRight="40dp"
11    android:paddingBottom="10dp"
12    tools:context=".MainActivity">
13
14    <TextView
15        android:id="@+id/textView"
16        android:layout_width="match_parent"
17        android:layout_height="wrap_content"
18        android:background="#03A9F4"
19        android:text="Welcome to MSc Final Mobile Development"
20        android:textAppearance="@style/TextAppearance.AppCompat.Large" />
21
22    <TextView
23        android:id="@+id/Result"
24        android:layout_width="match_parent"
25        android:layout_height="wrap_content"
26        android:text="Output"
```

Palette

LinearLayout

Design Text


```
    <Button  
        android:id="@+id/addButton"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:onClick=""  
        android:text="Add Values" />  
    </LinearLayout>
```

```
    <Button  
        android:id="@+id/addButton"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:onClick="addTwoValues"  
        android:text="Add Values" />
```

```
<Button
    android:id="@+id/addButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="addTwoValues"
    android:text="Add Values" />
</LinearLayout>
```

LinearLayout > Button

Text

/yApplication2

✖ Suppress: Add tools:ignore="OnClick" attribute

💡 Create 'addTwoValues(View)' in 'MainActivity' ▶

🔗 Create onClick event handler ▶

🔗 Override Resource in Other Configuration... ▶

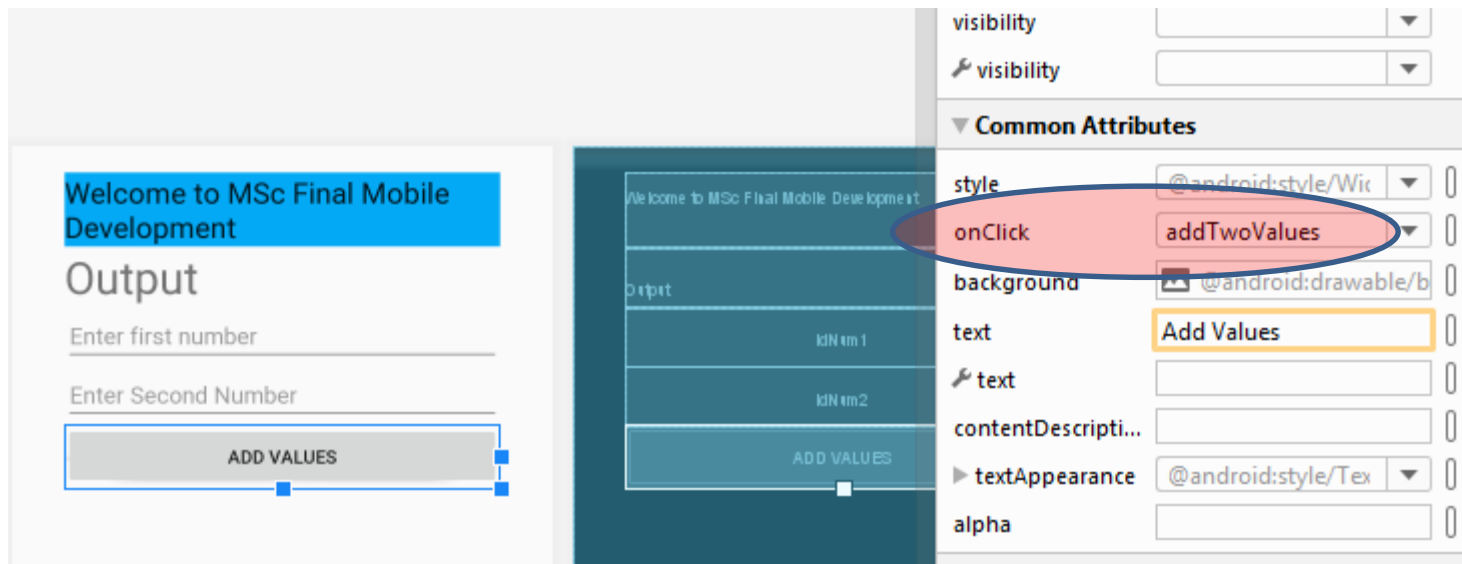
🔗 Rearrange tag attributes ▶

🔗 Remove attribute ▶

🔗 Inject language or reference ▶

```
1 package com.example.myapplication2;
2
3 import ...
4
5
6
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14     }
15
16     public void addTwoValues(View view) {
17     }
18 }
19
```

- Alternatively use **onClick**: property in the common attributes tab to create event handling function.



Accessing Widgets Values

- We need to access TextView and the two TextFields (EditText) in Java Button click handler.
- Accessing Text Field or EditText.
 `EditText obj = (EditText) findViewById (View-ID)`
- *View-ID is the ID assigned to the to the View or widget*
- *Use R.id.(**ID assigned to the View in the activity**)*
- For example: `R.id.Num1`
- Getting Value from the EditText.

Accessing and assigning text view values.

Text view is used as a text display (Read Only text) on activity.

TextView obj = findViewById (View-ID)

- No need to parse the object as we did for Edit Text.
- View-ID is the reference to the View or widget in the activity.
- Use R.id.(ID assigned to the View in the activity)
- E.g. **R.id.Num1**
- **Num1.getText().toString()** to get string.
- **Integer.parseInt()** is used to parse the value to string if the value is number.

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void addTwoValues(View view) {

        //create objects of the Edit Text fields to retrieve values
        EditText num1 = (EditText) findViewById(R.id.idNum1);
        EditText num2 = (EditText) findViewById(R.id.idNum2);

        TextView output = findViewById(R.id.Result);

        int value1 = Integer.parseInt(num1.getText().toString());
        int value2 = Integer.parseInt(num2.getText().toString());

        int total = value1+value2;

        output.setText(Integer.toString(total));
    }
}

```

Retrieve the two values using `getText()` function. We have to parse the values before adding.

```
public void addTwoValues(View view) {
```

```
    EditText num1 = (EditText) findViewById(R.id.)
```

```
    }  
}
```

f	idNum1	(= -1000114)	int
f	idNum2	(= -1000113)	int
f	addButton	(= -1000014)	int
f	gone	(= -1000072)	int
f	invisible	(= -1000004)	int
f	packed	(= -1000097)	int
f	parent	(= -1000000)	int
f	percent	(= -1000083)	int
f	Result	(= -1000059)	int
f	spread	(= -1000109)	int
f	spread_inside	(= -1000043)	int
f	textView	(= -1000041)	int

MainActivity > addTwoValues()

Access values in the text fields using **findViewById**


```
public void addTwoValues(View view) {  
  
    //create objects of the Edit Text fields to retrieve values  
    EditText num1 = (EditText) findViewById(R.id.idNum1);  
    EditText num2 = (EditText) findViewById(R.id.idNum2);  
  
    TextView output = findViewById(R.id.);  
}  
}
```

MainActivity > addTwoValues()

lyApplication2

f	idNum2	(= -1000113)	int
f	idNum1	(= -1000114)	int
f	addButton	(= -1000014)	int
f	gone	(= -1000072)	int
f	invisible	(= -1000004)	int
f	packed	(= -1000097)	int
f	parent	(= -1000000)	int
f	percent	(= -1000083)	int
f	Result	(= -1000059)	int
f	spread	(= -1000109)	int
f	spread_inside	(= -1000043)	int
f	textView	(= -1000041)	int

Access the textView to display the sum of the two values using **findViewById**

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void addTwoValues(View view) {

        //create objects of the Edit Text fields to retrieve values
        EditText num1 = (EditText) findViewById(R.id.idNum1);
        EditText num2 = (EditText) findViewById(R.id.idNum2);

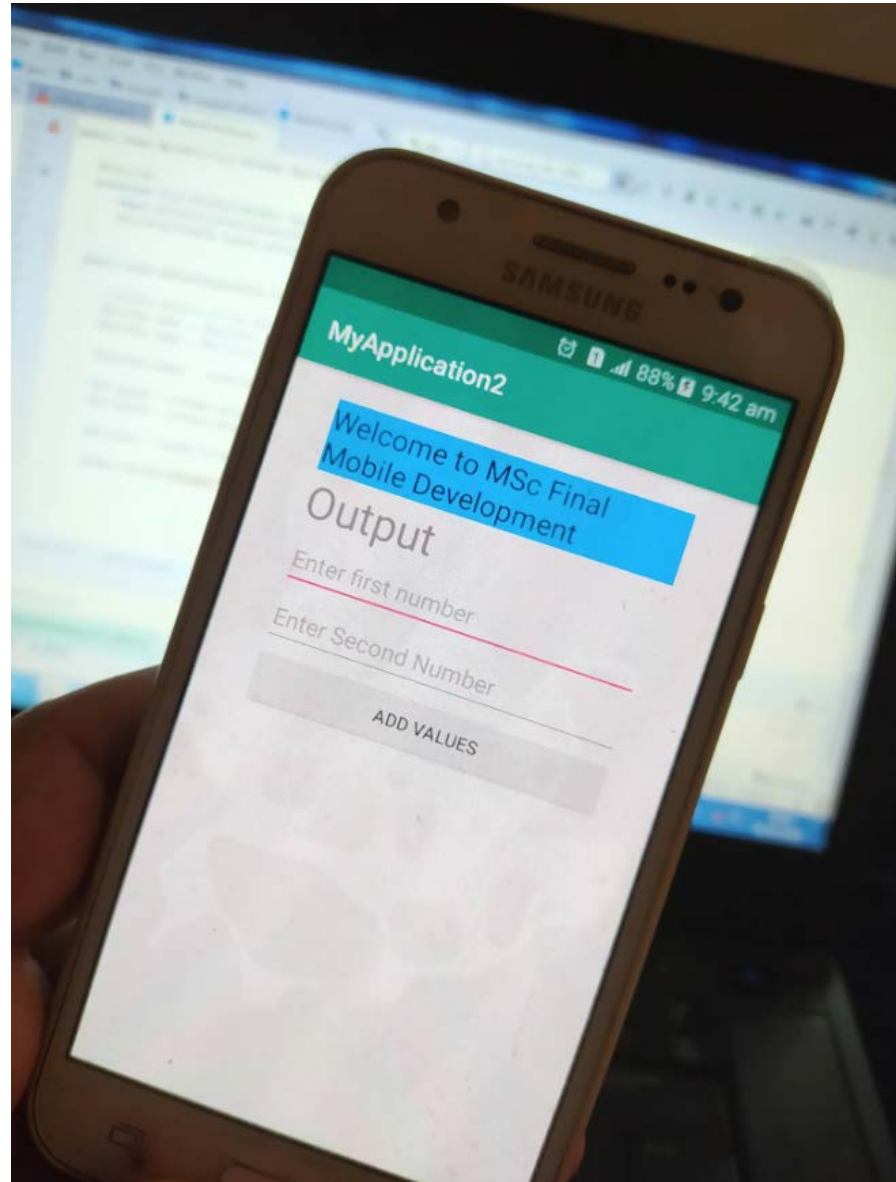
        TextView output = findViewById(R.id.Result);

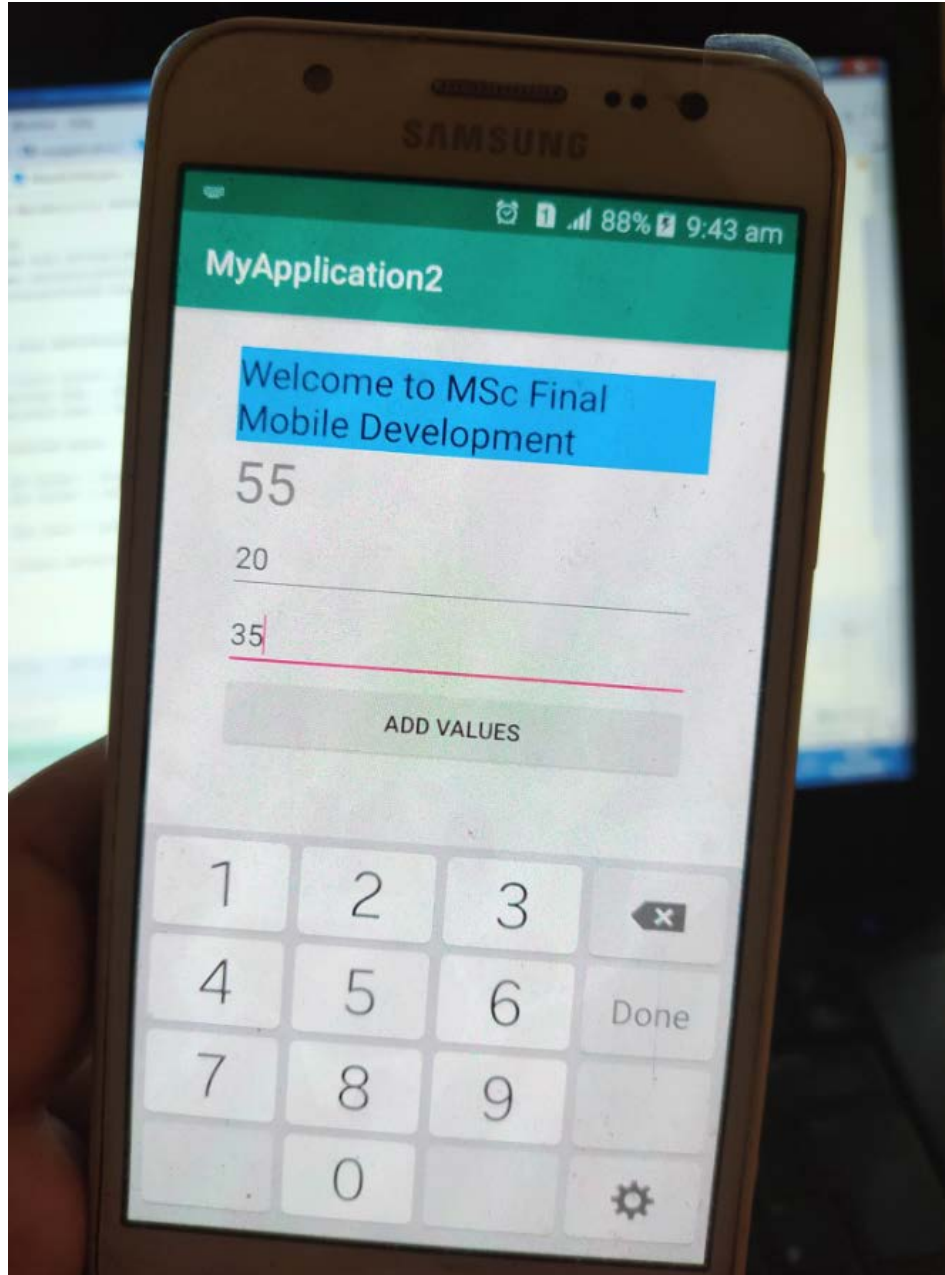
        int value1 = Integer.parseInt(num1.getText().toString());
        int value2 = Integer.parseInt(num2.getText().toString());

        int total = value1+value2;

        output.setText(Integer.toString(total));
    }
}
```

Retrieve the two values using `getText()` function. We have to parse the values before adding.





activity_main.xml x MainActivity.java x

Palette Q ⚙ — 🔍 🔧 🔍 🔧 — Pixel 29 AppTheme Default (en-us) 28% 🔍 ⚙ —

Common **TextView**
 Plain Text
 Password
 Password (Num...
 E-mail
 Phone
 Postal Address
 Multiline Text
 Time
 Date
 Number
 Number (Signed)
 Number (Decim...
 AutoCompleteT...
 MultiAutoComp...
 CheckedTextView
 TextInputLayout

Component Tree ⚙ —

- LinearLayout(vertical)
 - textView- "Welcome to MSC ..."
 - Result- "Output"
 - idNum1 (Number)**
 - idNum2 (Number)
 - addButton- "Add Values"

Attributes Q ⚙ —

idNum1 EditText

id idNum1

Declared Attributes + —

Layout

Common Attributes

inputType number

hint Enter first number

style @style/Widget.App

singleLine

selectAllOnFocus

text

text

contentDescripti...

textAppearance @android:style/Tex

alpha

All Attributes

alpha

autoLink

autoText

background @android:drawable/e

bufferType

capitalize

clickable true

contentDescriptio

cursorVisible

digits

drawableBottom

drawableEnd

drawableLeft