Buse Çarık 27/06/18

INTERRA R&D Material Design

Material Design

Button

https://github.com/InterraMaterialDesign/MaterialButton

**INDEX**

1. **INTRODUCTION……………………………………………………………………….3**
   1. **Purpose of this document…………………………………………………………...3**
   2. **Overview……………………………………………………………………………..3**
2. **WHAT IS BUTTON…..…………………………………………………………………4**
   1. **Material Design……………………………………………………………………...6**
3. **REQUIREMENTS FOR IMPLEMENTATION………………………………………8**
4. **AREAS OF USAGE……………………………………………………………………...8**
5. **MATERIAL BUTTON…..………………………………………………………………9**
   1. **Styles………………………………………………………………………………….9**
   2. **Animation…………………………………………………………………………..10**
   3. **Attributes…………………………………………………………………………...10**
6. **REFERENCES………………………………………………………………………….29**

**INTRODUCTION**

**PURPOSE OF THIS DOCUMENT**

The aim of this project is analyzing material design button’s styles and attributes.

**OVERVIEW**

The project that is described in this document, includes material button’s types and styles. The types are filled elevated button, filled unelevated button and floating action button and the styles are

Widget.MaterialComponents.Button, Widget.MaterialComponents.Button.UnelevatedButton, Widget.MaterialComponents.Button.TextButton, TextAppearance.MaterialComponents.MaterialButton, Widget.MaterialComponents.Button.TextButton, Widget.MaterialComponents.Button.UnelevatedButton, TextAppearance.MaterialComponents.MaterialButton.Secondary, Widget.MaterialComponents.FloatingActionButton

**Button**

Buttons allows users to perform actions and make choices with a tap or click. It extends from TextView. A button can be made of text, icon or both. If there is only text or text and icon are present together, basic button is used. If the button consists of only an icon, ***ImageButton*** is used.

<ImageButton  
    android:layout\_width="wrap\_content"  
    android:layout\_height="wrap\_content"  
    android:src="@drawable/button\_icon"  
    ... />

<Button  
    android:layout\_width="wrap\_content"  
    android:layout\_height="wrap\_content"  
    android:text="@string/button\_text"  
    ... />

*Button with only text Button with only icon*

<Button  
    android:layout\_width="wrap\_content"  
    android:layout\_height="wrap\_content"  
    android:***text***="@string/button\_text"  
    android:***drawableLeft***="@drawable/button\_icon"  
    ... />

*Button with both icon and text*

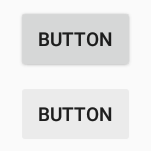
**ImageButton**

It extends from ImageView. When it is pressed or clicked, it performs an action. It looks like a regular button; the difference is that only an icon or image is placed instead of text. You can customize background image with ***android:src*** attribute. Also, you can place different images for different states like when the button is pressed or hovered.

**Styling the Button**

Button’s background can be customized using ***android:background*** attributewith drawable or color resource. The other way is applying ***style*** for that button. One of the present style is ***borderless button***. To apply this:

<Button   
    android:layout\_width="wrap\_content"  
    android:layout\_height="wrap\_content"  
    android:text="@string/button"  
    style="***?android:attr/borderlessButtonStyle***" />



*Default borderless button Pressed borderless button*

Also, you can customize your button’s background for different **states**:

First, each states’ images are placed into a res/drawable/ files. Then, create a new res/drawable file and apply your file so that the following lines will be suitable for your project:

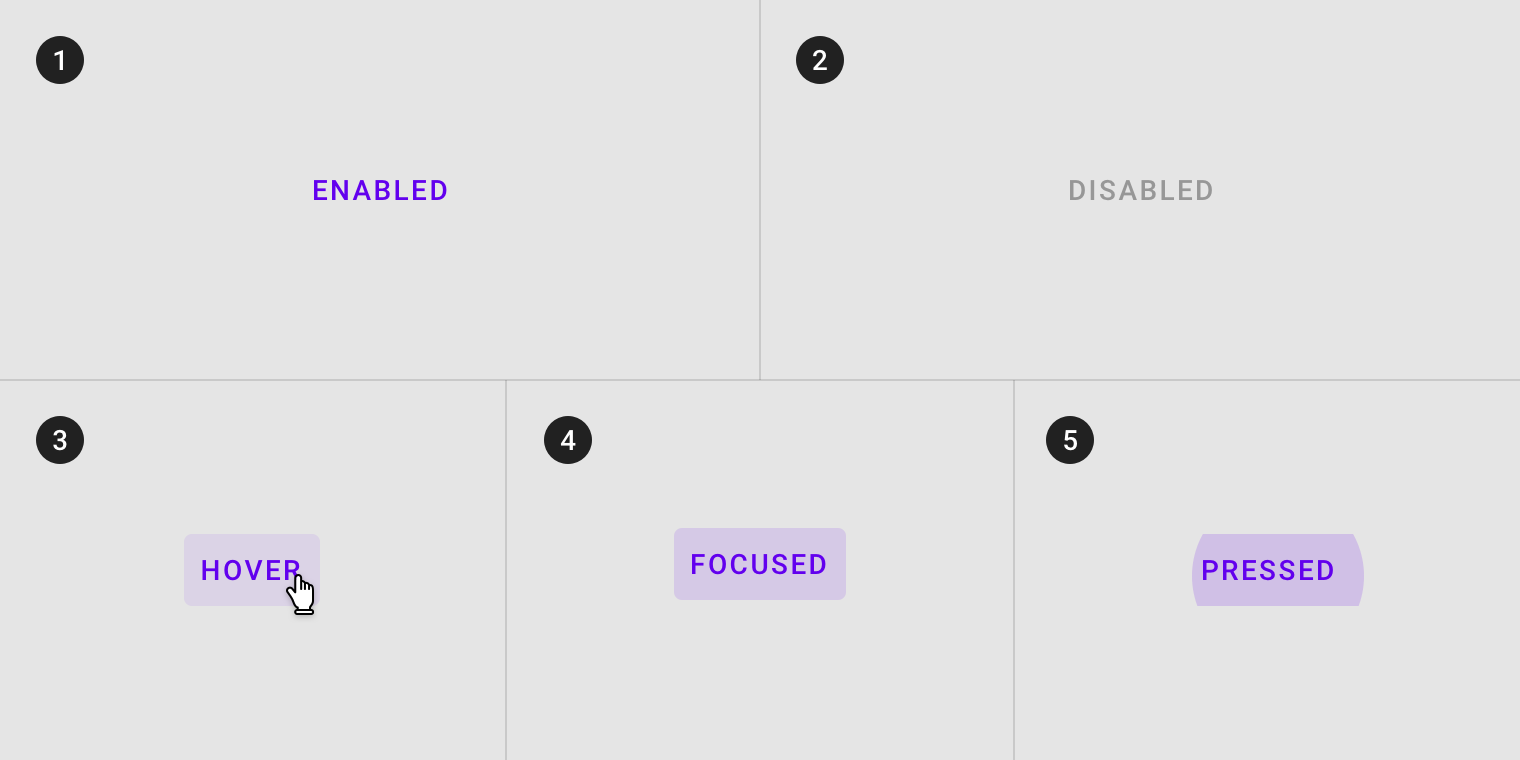
<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
    <item android:drawable="@drawable/button\_pressed"  
          android:state\_pressed="true" />  
    <item android:drawable="@drawable/button\_focused"  
          android:state\_focused="true" />  
    <item android:drawable="@drawable/button\_default" />  
</selector>

**Material Design**

According to Material Design, the text label should be a single line text. And by default, the entire text is written in uppercase. In material design, there are four types of buttons:

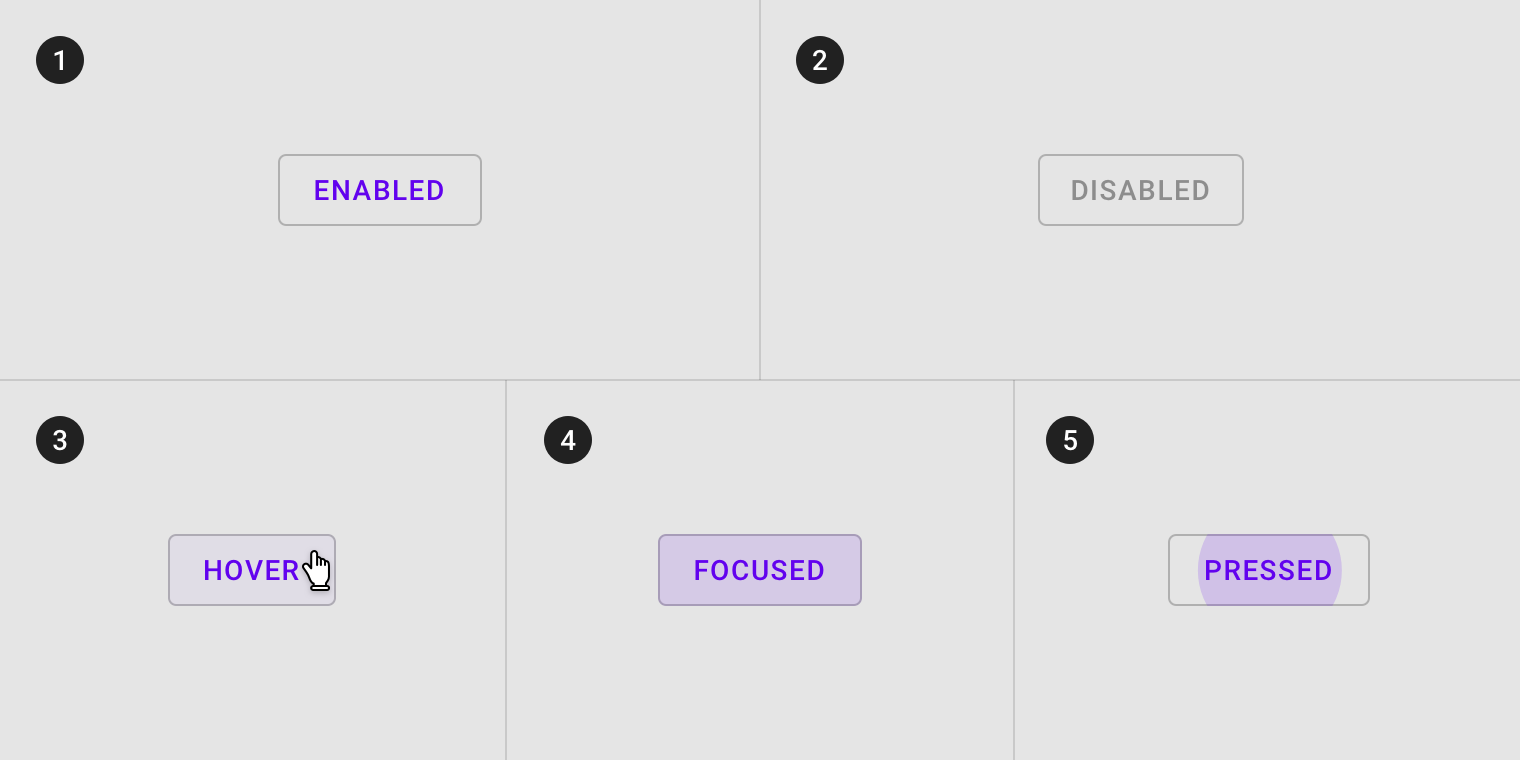
1. **Text Button**

This button is used for low emphasis actions. They are used in dialogs, cards and snackbars. **States**:



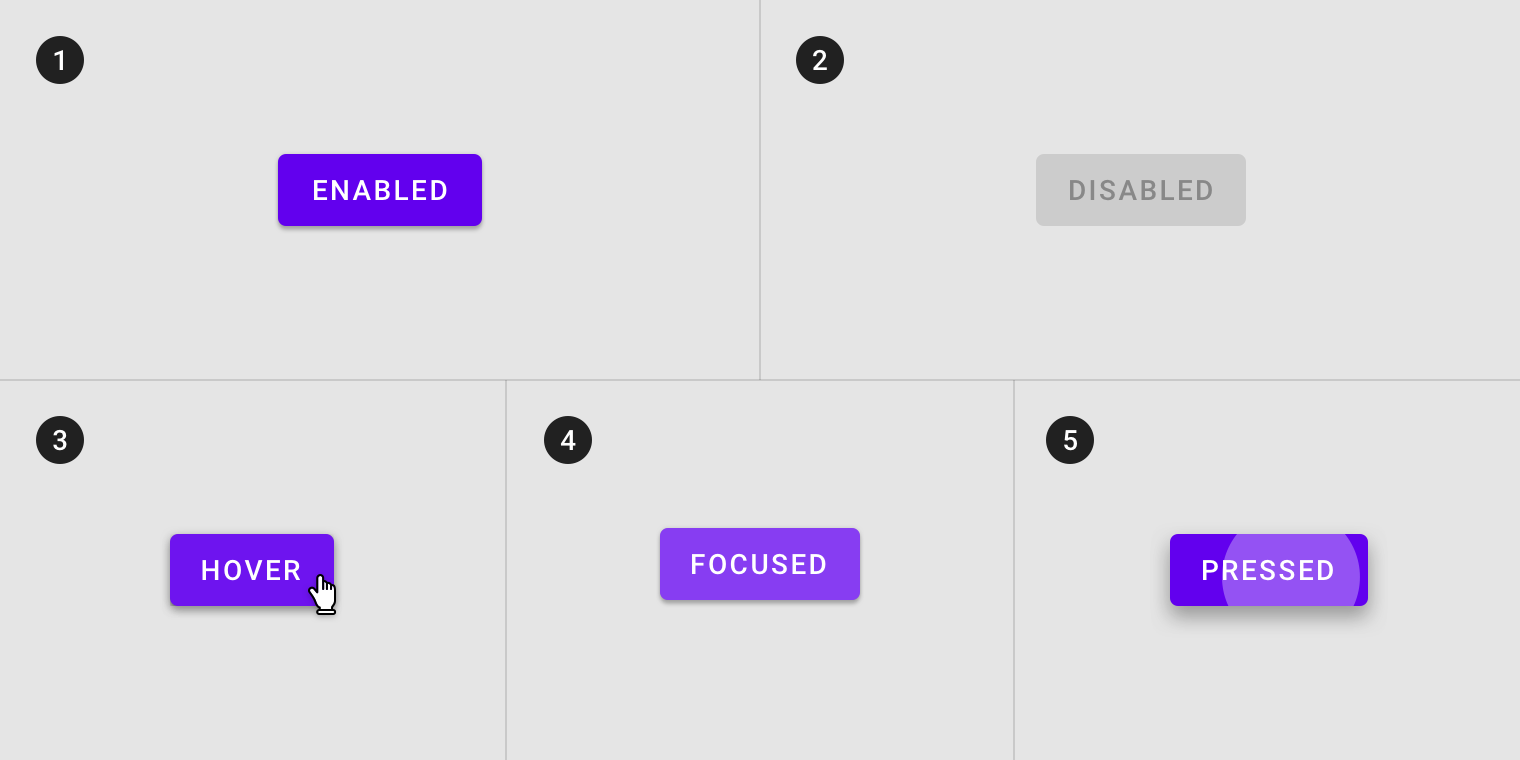
1. **Outlined Button**

This button is used for medium emphasis actions. These actions are important but not primary actions. The main feature of this button type is stroke. Its background color is transparent and the differences between text button and outlined button is stroke. **States**:



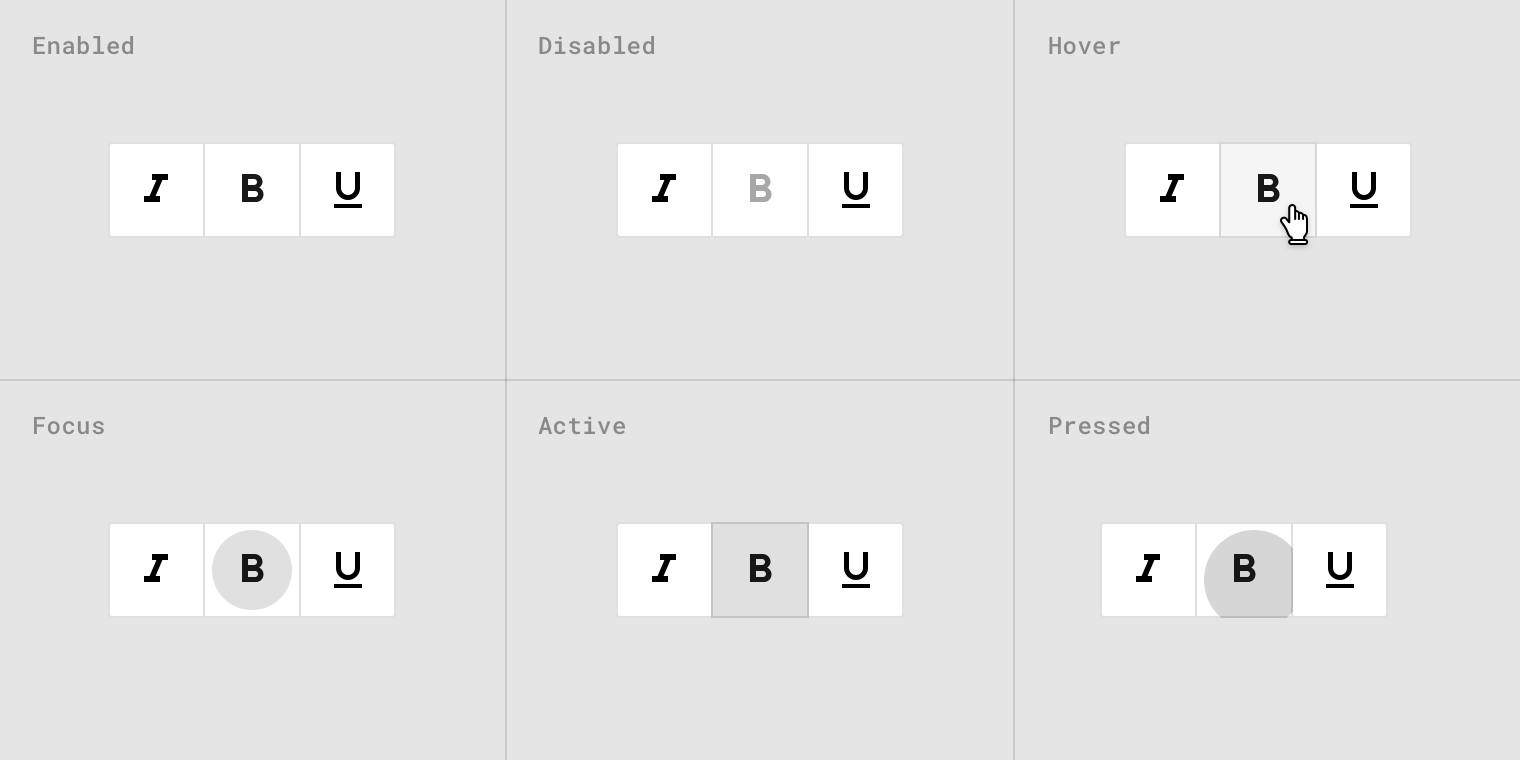
1. **Contained Button**

This button type is used for primary, high emphasis actions. They have a container with a colored background and have shadow which is set by ***elevation*** attribute. **States:**



1. **Toggle Button**

This button type is used for grouping a set of actions. Related toggle buttons should be in the same container to provide emphasis. You can select only one toggle button in a group at a time. There is no toggle button implementation for the Material Design. **States:**



**Requirements for Implementation**

To use Material Design’s button, add the following requirement to the build.gradle file in the project:

dependencies {  
 implementation 'com.android.support: design:28.0.0-alpha1 }

In XML file, button is called as Button. If the Material Design’s button is preferred, android.support.design.button.MaterialButton should be used.

**Areas of Usage**

Buttons are used in dialogs, modal windows, forms, cards and toolbars to communicate actions. Their places are changed according to buttons’ type. For example, text buttons cannot distinguish easily because they do not have a container, therefore they are placed in contained components like cards or dialogs.

**Material Button**

Material button which extends from ***AppCompatButton,*** has different styles to customize. There are two types of buttons according to the emphasis level:

**Raised button**: when the button is pressed, it elevates. colorAccent is default color for background. Text color is white.

**Unelevated button**: when the button is pressed, its elevation does not change. Its background is transparent, text color id colorAccent.

There are different ***styles*** for material button:

* **Filled, elevated button**: it has colored background and elevation. This button style should use in high important actions. (default style) Background color is primary color by default.
* **Filled, unelevated button**: unelevated button with colored background (primary color).
* **Text button**: transparent background, colored text. Use for low important actions.
* **Floating action button**: provides a quick access to important or most used actions

In style resource file, there are several existing styles for material button. While a custom style was creating, you must extend one of those existing styles as parent.

**Styles**

* Widget.MaterialComponents.Button
* TextAppearance.MaterialComponents.MaterialButton (for text in that button)
* Widget.MaterialComponents.Button.TextButton
* Widget.MaterialComponents.Button.UnelevatedButton
* TextAppearance.MaterialComponents.MaterialButton.Secondary (for text in that button)
* Widget.MaterialComponents.FloatingActionButton

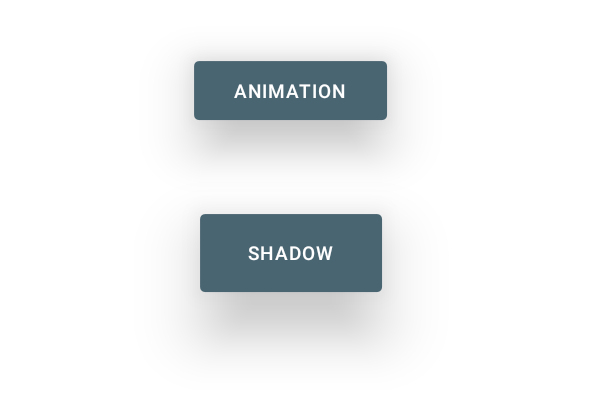
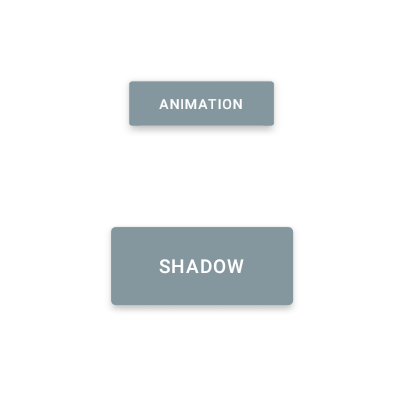
**Attributes**

* ***backgroundTint***: set a tint(tone) to the background of the button.
* ***icon***: set an icon into the button
* ***additionalPaddingStartForIcon***: add extra space between border of button and left of icon
* ***additionalPaddingEndForIcon***: add extra space between border of button and right of icon
* ***strokeColor***: change the color of the border of the button.
* ***strokeWidth***: change the thickness of the border of the button.
* ***cornerRadius***: set the roundness of the corners of button
* ***rippleColor***: it changes the background color of button while the button is pressed
* ***iconPadding***: arrange space between icon and its surroundings
* ***iconTint***: set the icon’s tint
* ***stateListAnimator***: set an animation when the button’s situation is changed

**Animation**

With ***Animator*** resource file, animations can be set to the view. To set, ***stateListAnimator*** feature is used in XML file. In the selector option, different states can be found. With ***objectAnimator*** subclass, objects’ properties can be changed in different states.

For example, their scale can be increased, when the button is pressed or with ***translationZ*** attribute, depth sensation can be created.

shadow\_animation.xml

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

<item android:state\_pressed="true">

<set**>**

<objectAnimator

android:duration="0"

android:propertyName="elevation"

android:valueTo="2dp"

android:valueType="floatType" />

<objectAnimator

android:duration="250"

android:propertyName="translationZ"

android:valueTo="2dp"

android:valueType="floatType" />

</set>

</item>

<item>

<set>

<objectAnimator

android:duration="@android:integer/config\_shortAnimTime"

android:propertyName="elevation"

android:valueTo="6dp"

android:valueType="floatType" />

<objectAnimator

android:duration="@android:integer/config\_shortAnimTime"

android:propertyName="translationZ"

android:valueTo="30dp"

android:valueType="floatType" />

</set>

</item>

</selector>

animation.xml

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

<item android:state\_pressed="true">

<set android:ordering="together"**>**

<objectAnimator

android:duration="250"

android:propertyName="scaleX"

android:valueTo="0.75"

android:valueType="floatType" />

<objectAnimator

android:duration="250"

android:propertyName="scaleY"

android:valueTo="0.75"

android:valueType="floatType" />

<objectAnimator

android:duration="250"

android:propertyName="translationZ"

android:valueTo="5dp"

android:valueType="floatType" />

</set>

</item>

<item>

<set>

<objectAnimator

android:duration="@android:integer/config\_shortAnimTime"

android:propertyName="scaleY"

android:valueTo="1.0"

android:valueType="floatType" />

<objectAnimator

android:duration="@android:integer/config\_shortAnimTime"

android:propertyName="scaleX"

android:valueTo="1.0"

android:valueType="floatType" />

<objectAnimator

android:duration="@android:integer/config\_shortAnimTime"

android:propertyName="translationZ"

android:valueTo="30dp"

android:valueType="floatType" />

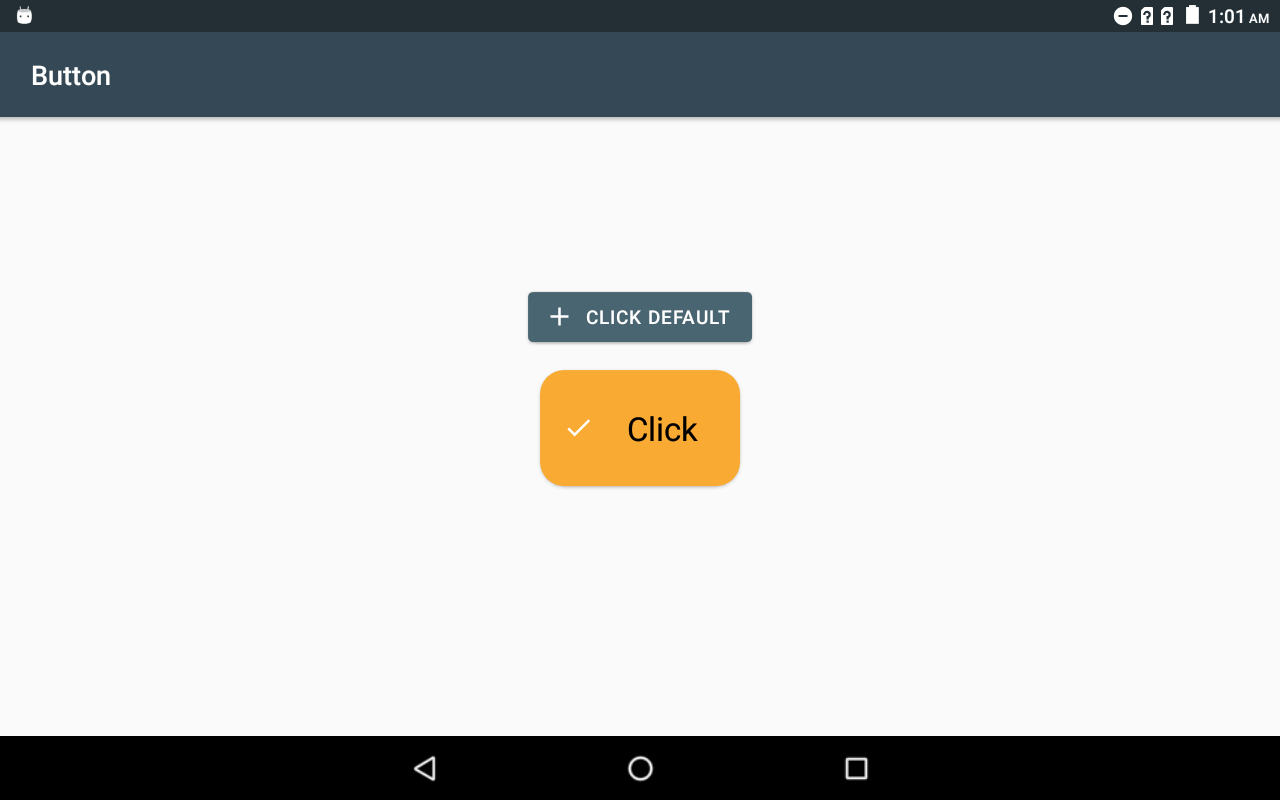
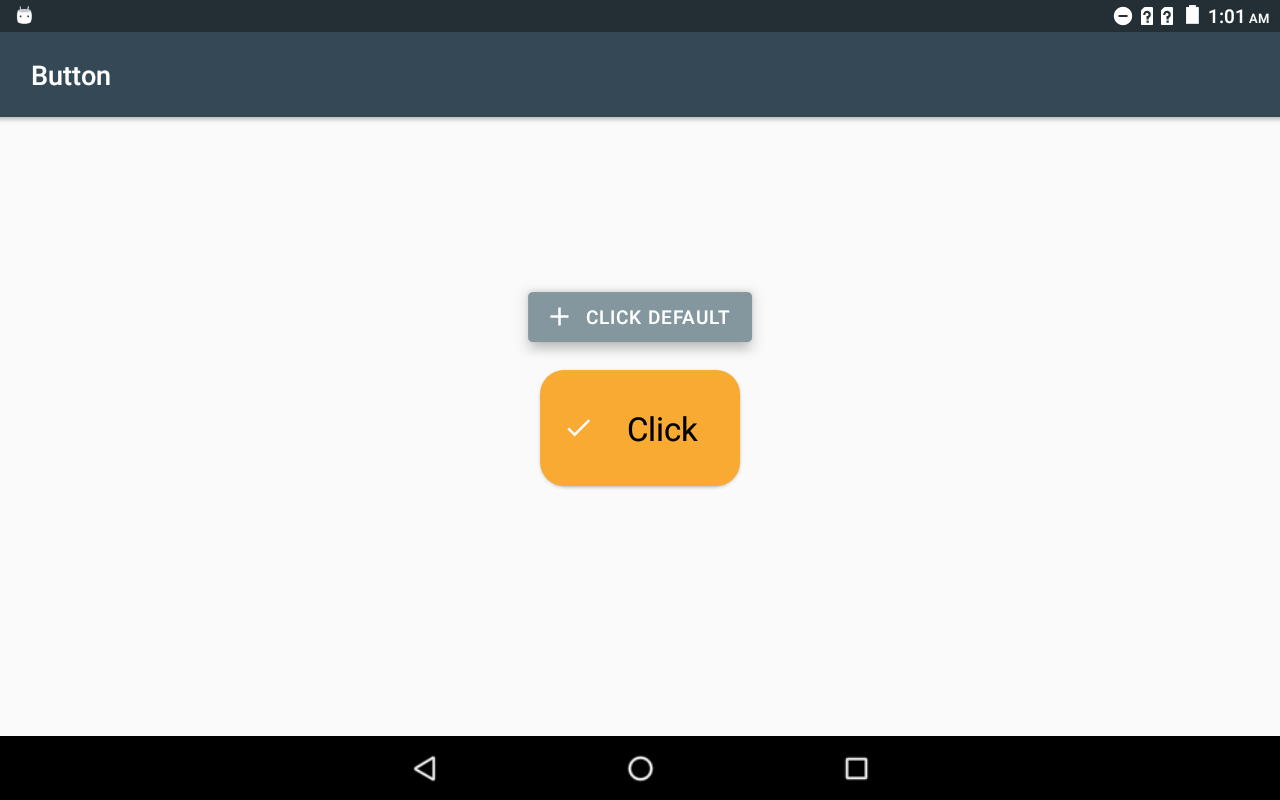
</set>

</item>

</selector>

**Widget.MaterialComponents.Button**

Its parent is Widget.AppCompat.Button. Its customized attributes are ***backgroundTint***, ***rippleColor***, ***cornerRadius***, ***insetLeft***, ***insetRight, insetTop, insetBottom, textAppearance, iconTint, iconPadding, stateListAnimator, additionalPaddingLeftForIcon, additionalPaddingRightForIcon.***

The click default is default settings of material button. In the second button which is customized.

Custom style

<style name="materialButton" parent="Widget.MaterialComponents.Button" >

<item name="cornerRadius">18dp</item>

<item name="backgroundTint"> #F9AA33</item>

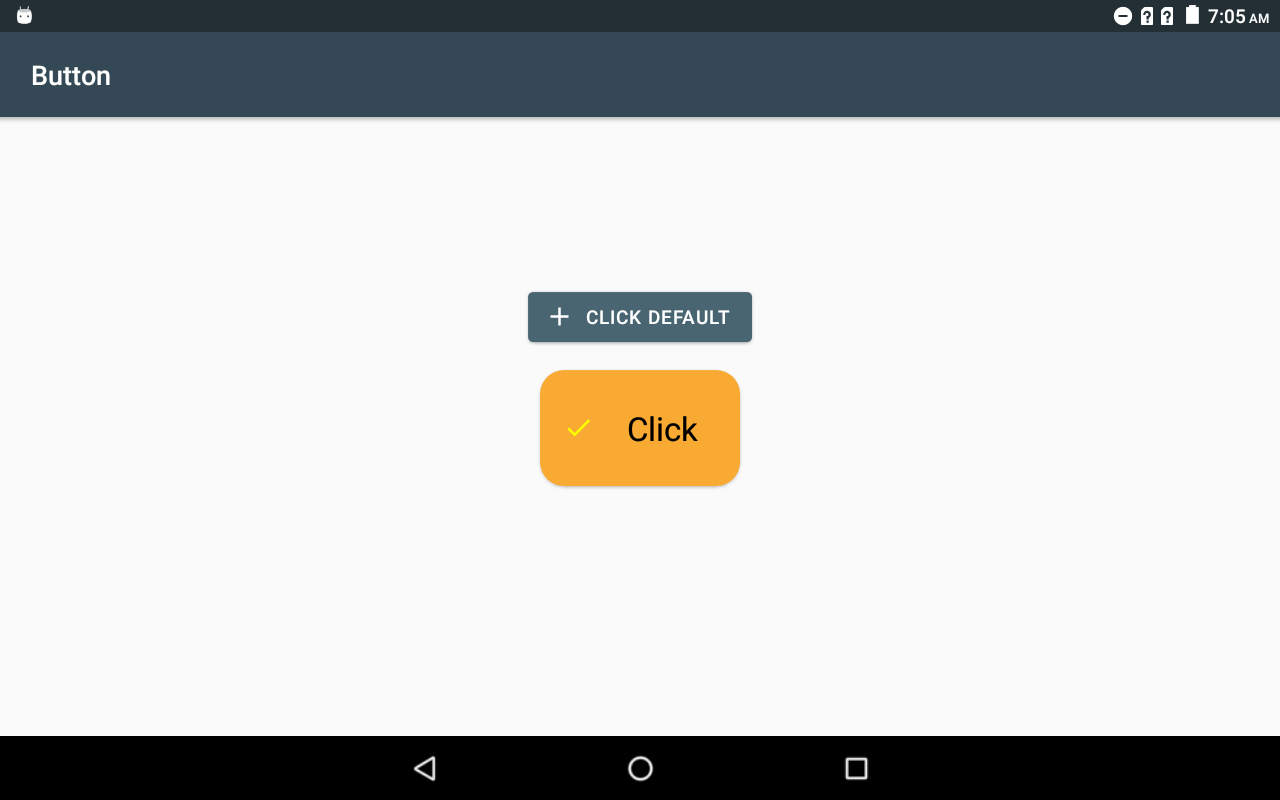
<item name="iconPadding">8dp</item>

<item name="additionalPaddingRightForIcon">2dp</item>

<item name="android:focusable">true</item>

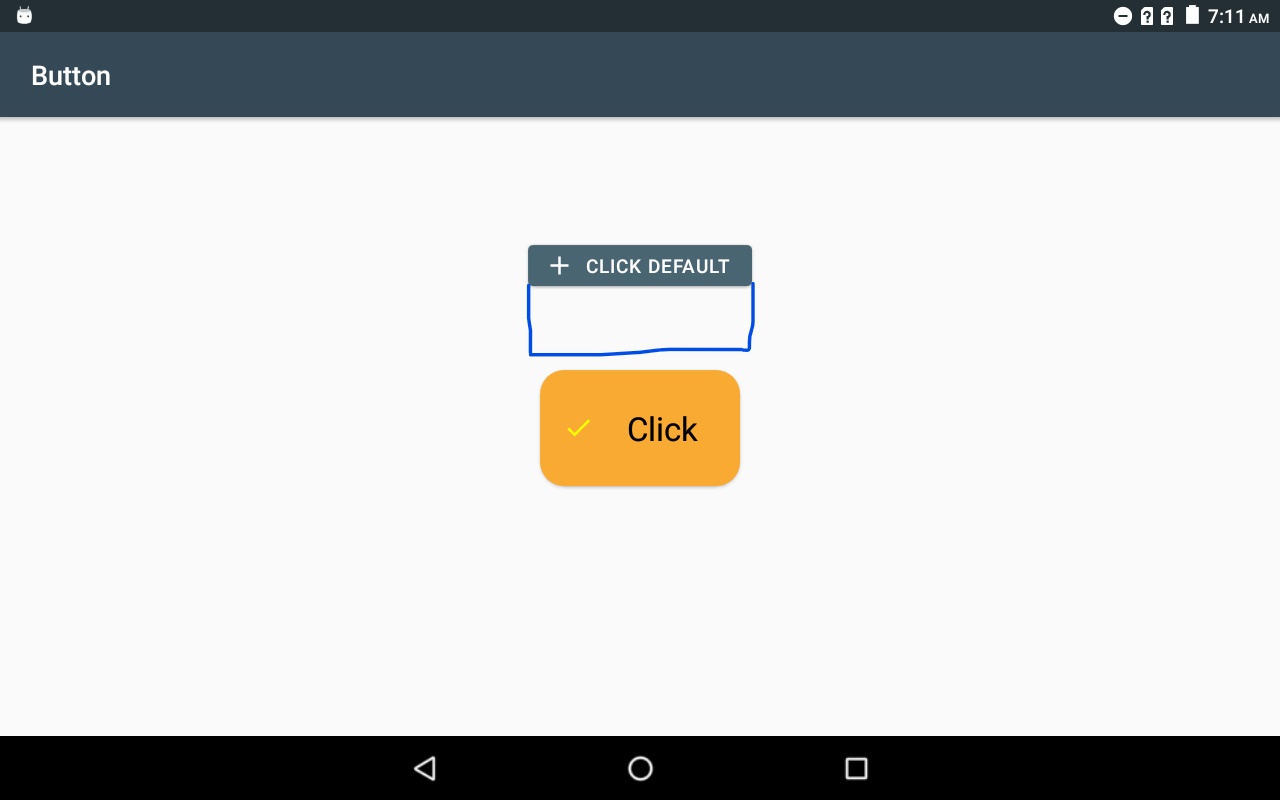
<item name="android:insetTop">8dp</item>

</style>



Make icon yellow

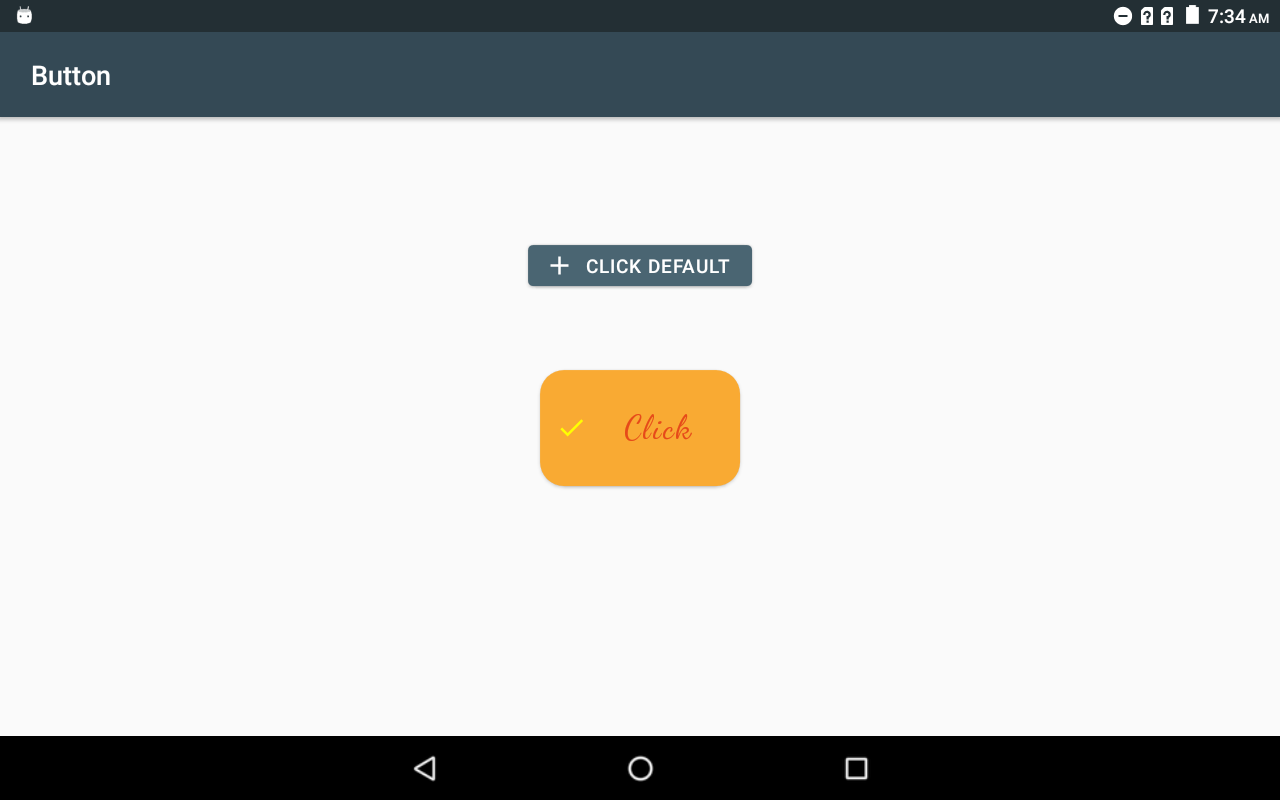
<item name="iconTint"> #FFFF00</item>



<item name="android:insetBottom">8dp</item>



<item name="rippleColor">#388E3C</item>



<item name="android:textAppearance">@style/text\_style\_button</item>

<style name="text\_style\_button" parent="TextAppearance.MaterialComponents.MaterialButton">

<item name="fontFamily">cursive</item>

<item name="android:textSize">30sp</item>

<item name="android:textAllCaps">false</item>

<item name="android:textColor">#E64A19</item>

<item name="android:gravity">center</item>

</style>

**TextAppearance.MaterialComponents.MaterialButton.Secondary**

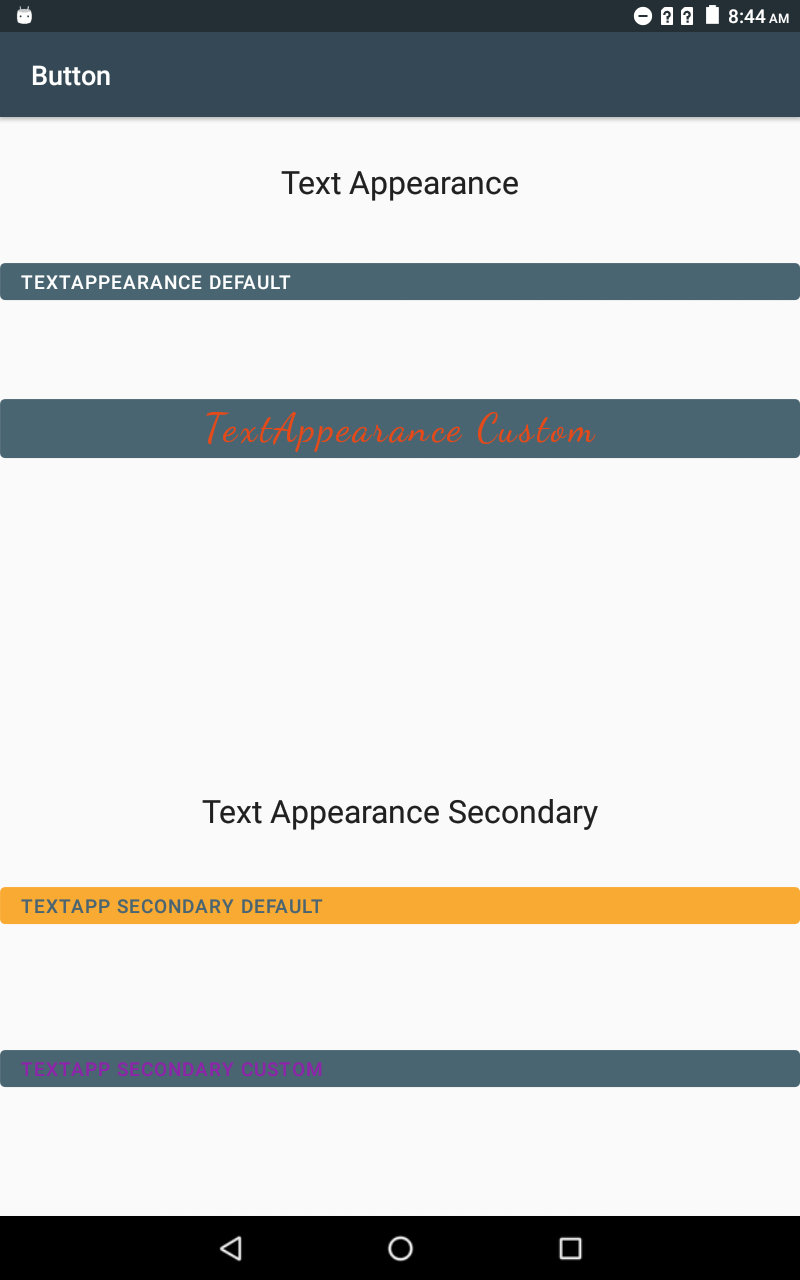
This style only customizes the text color attribute as a difference. Other attributes are the same with view attributes. This style does not contain material design button attributes. In default, ***textColor*** is set as ***colorAccent.***

**TextAppearance.MaterialComponents.MaterialButton**

This style’s parent is ***android:textAppearance***. Its attributes are ***textColor, textColorHighlight, textColorHint, textColorLink, textSize, textStyle***(bold, italic, normal). Material button text appearance customize textColor, textAllCaps, textSize, fontFamily(monospace, casual, cursive, sans-serif, sans-serif-condensed, serif, serif-monospace, sans-serif-smallcaps) and letterSpacing. This style does not have material design button attributes.

In fontFamily attribute, this style creates a new element which is sans-serif-medium(default).

In default, clickable attribute is false.



**TextApperance Default Style**

<style name="TextAppearance.MaterialComponents.MaterialButton" parent="android:TextAppearance">

<item name="android:textColor”>#ffffffff</item>

<item name="android:textAllCaps”>true</item>

<item name="android:textSize">14sp</item>

<!-- Roboto medium -->

<item name="fontFamily" ns1:ignore="NewApi">sans-serif-medium</item>

<item name="android:letterSpacing" ns1:ignore="NewApi">0.07 </item>

</style>

**TextApperance Custom Style**

<style name="text\_style\_button" parent="TextAppearance.MaterialComponents.MaterialButton">

<item name="fontFamily">cursive</item>

<item name="android:textSize">30sp</item>

<item name="android:textAllCaps">false</item>

<item name="android:textColor"># E64A19</item>

<item name="android:gravity">center</item>

</style>

android.support.design.button.MaterialButton’s default background color is colorAccent. TextApp Secondary Default’s style is TextAppearance.MaterialComponents.MaterialButton. Secondary. Secondary text appearance has only one custom attributes which is ***textColor***. And the default text color is ***colorAccent***. So, when we use android.support.design.button.MaterialButton, the background color is also become colorAccent, as a result, the text is not visible. So, in that example, ***backgroundTint*** attribute set as #F9AA33.

**TextApperance Secondary Default Style**

<style name="TextAppearance.MaterialComponents.MaterialButton.Secondary">

<item name="android:textColor">colorAccent</item>

</style>

XML file

android:backgroundTint="#F9AA33"

**TextApperance Secondary Custom Style**

<style name="text\_appearance\_secondary\_button" parent="TextAppearance.MaterialComponents.MaterialButton.Secondary">

<item name="android:textColor"> #8E24AA</item>

</style>

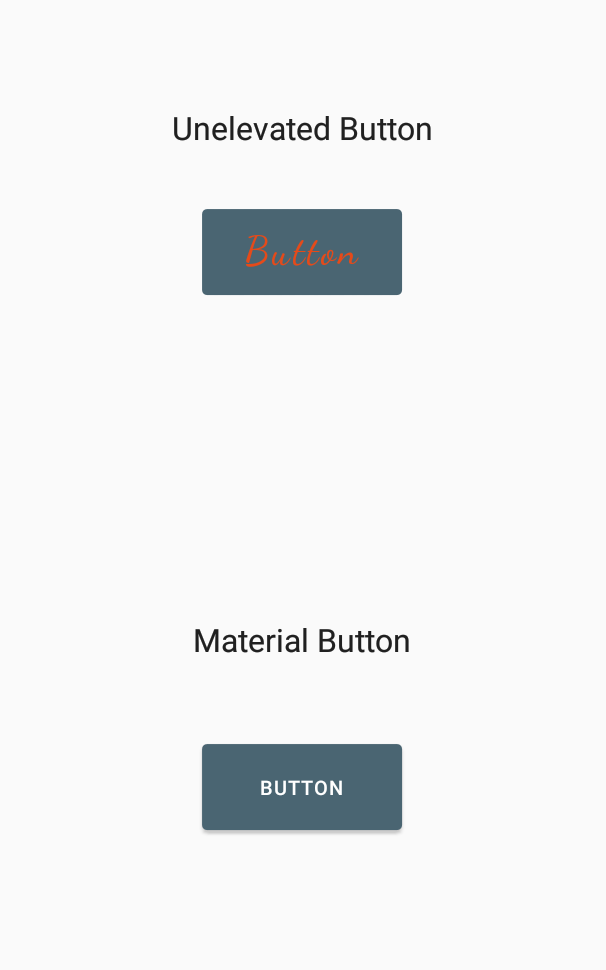
**Widget.MaterialComponents.Button.UnelevatedButton**

This style customizes only stateListAnimator attribute. The button has not elevation and when the button is pressed, its translationZ value does not change.

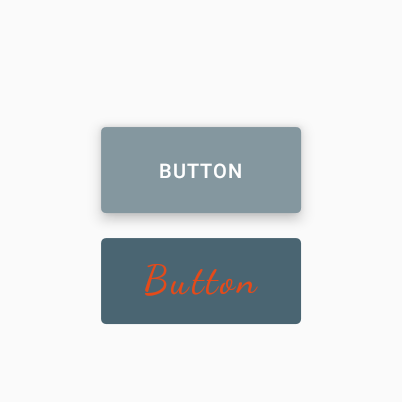
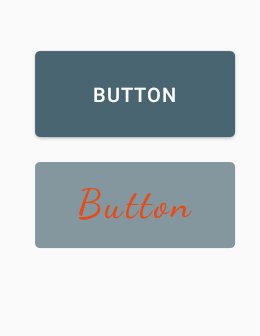
<style name="Widget.MaterialComponents.Button.UnelevatedButton">

<item name="android:stateListAnimator" ns1:ignore="NewApi">@animator/mtrl\_btn\_unelevated\_state\_list\_anim</item>

</style>



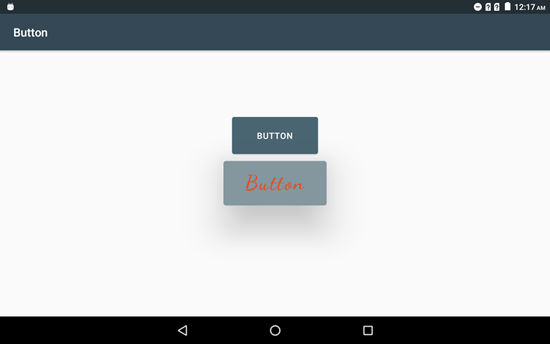
The button with white text is default material button and the button with red text’s style extends from Widget.MaterialComponents.Button.UnelevatedButton.

(material button is pressed) (material unelevated button is pressed)

In material button, when the button is pressed, its elevation increases, and it is felt that the button become closer. But in default unelevated button, when the button is pressed, the elevation does not change.

The stateListAnimator attribute can be customized also in unelevated button with animator resource file:



**Unelevated Button Custom**

<style name="unelevated\_button" parent="Widget.MaterialComponents.Button.UnelevatedButton">

<item name="android:stateListAnimator">@animator/animator</item>

<item name="android:textAppearance" >@style/text\_style\_button</item>

</style>

**Widget.MaterialComponents.Button.TextButton**

This style’s parent is ***Widget.MaterialComponents.Button.UnelevatedButton***. Its default background color is transparent therefore you cannot observe the clickable and focusable attributes. Background color cannot be changed but you can set backgroundTint color which make enable to observe clickable and focusable attributes. Besides, with textAppearance you can add a style to change text’s features.

Its customized attributes: ***backgroundTint, iconPadding, iconTint, rippleColor, paddingRight, paddingLeft, textAppearance, stateListAnimator***(unelevatedButton’s attribute)

Text Button Default

</style>

<style name="Widget.MaterialComponents.Button.TextButton" parent="Widget.MaterialComponents.Button.UnelevatedButton">

<item name="android:textAppearance">@style/TextAppearance.MaterialComponents.MaterialButton.Secondary</item>

<item name="android:paddingLeft">8dp</item>

<item name="android:paddingRight”>8dp</item>

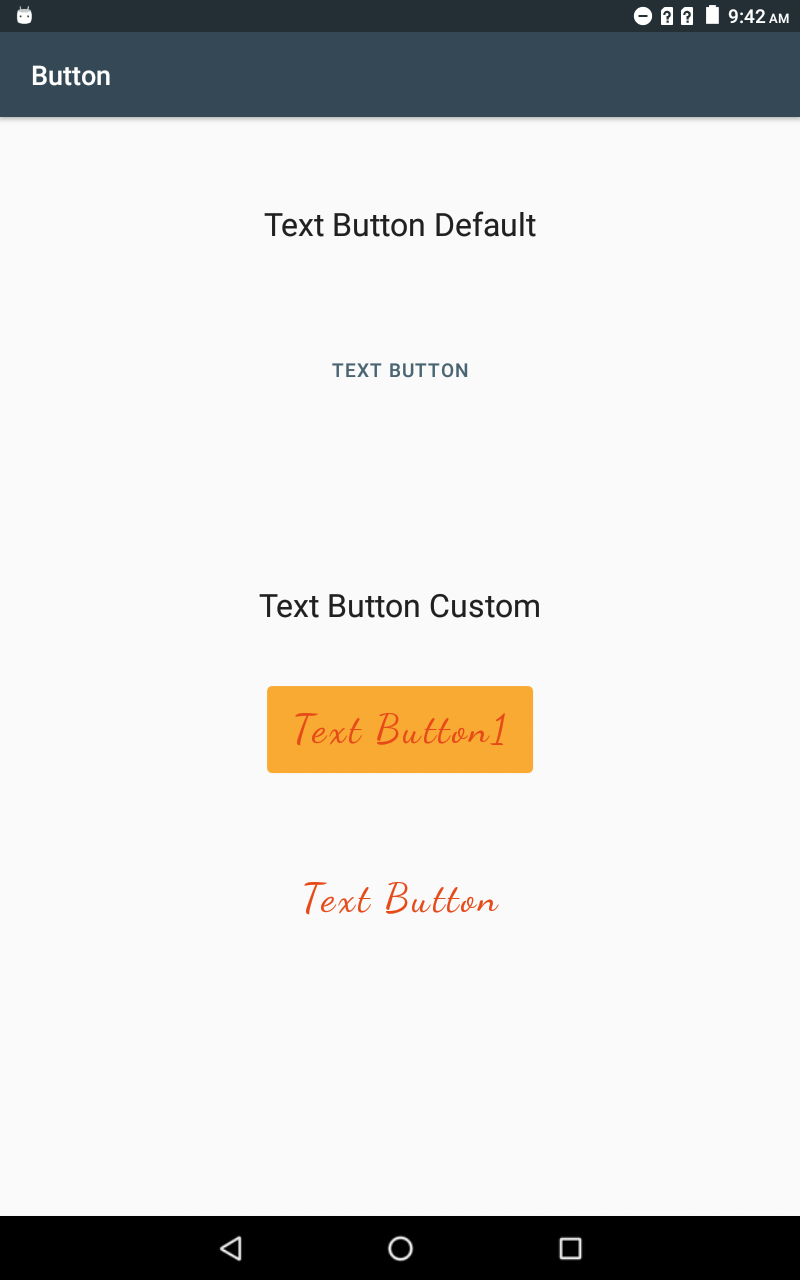
<item name="iconTint">@color/colorAccent</item>

<item name="iconPadding">4dp</item>

<item name="backgroundTint"> #00ffffff</item>

<item name="rippleColor">@color/colorAccent</item>

</style>



Text Button1 Custom Style

<style name="text\_button\_backgroundTint" parent="Widget.MaterialComponents.Button.TextButton">

<item name="rippleColor"> #FF4081</item>

<item name="android:textAppearance">@style/text\_style\_button</item>

<item name="backgroundTint"> #F9AA33</item>

</style>

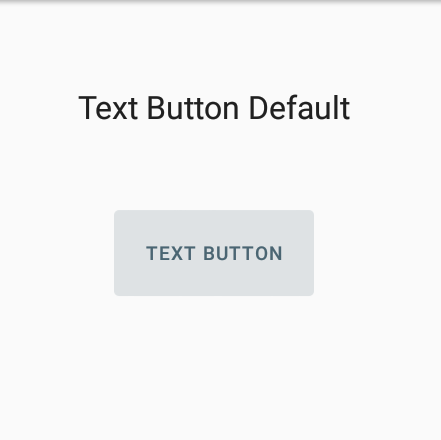
Text Button Custom Style

<style name="text\_button" parent="Widget.MaterialComponents.Button.TextButton">

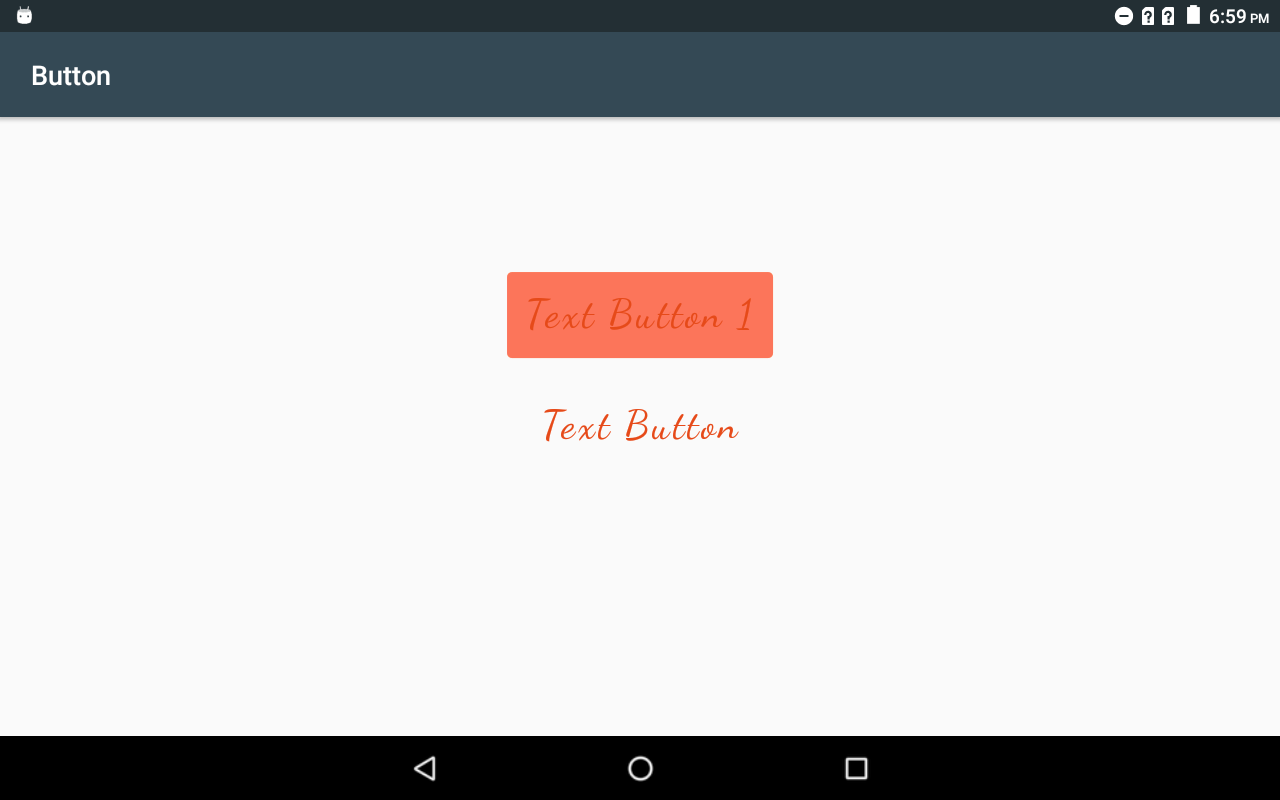
<item name="rippleColor"> #FF4081</item>

<item name="android:textAppearance">@style/text\_style\_button</item>

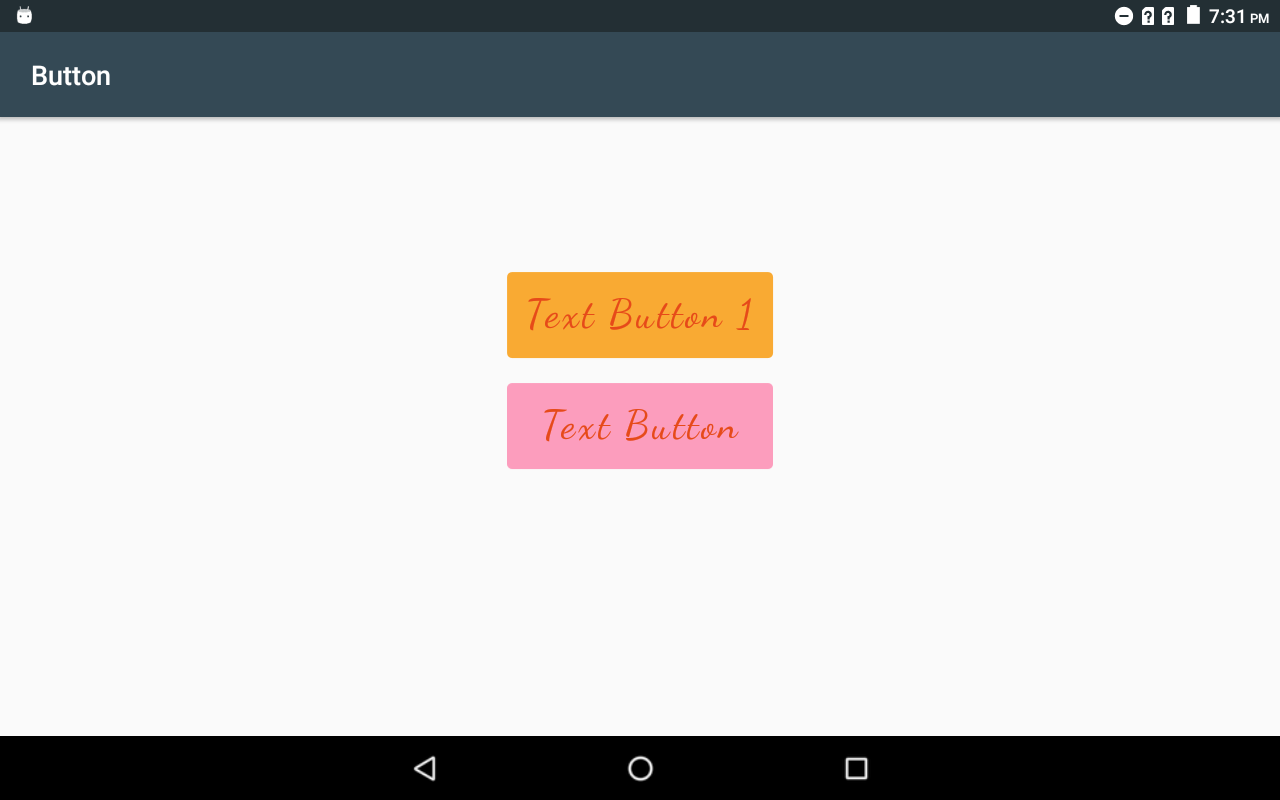
</style>



Text Button Default is pressed.



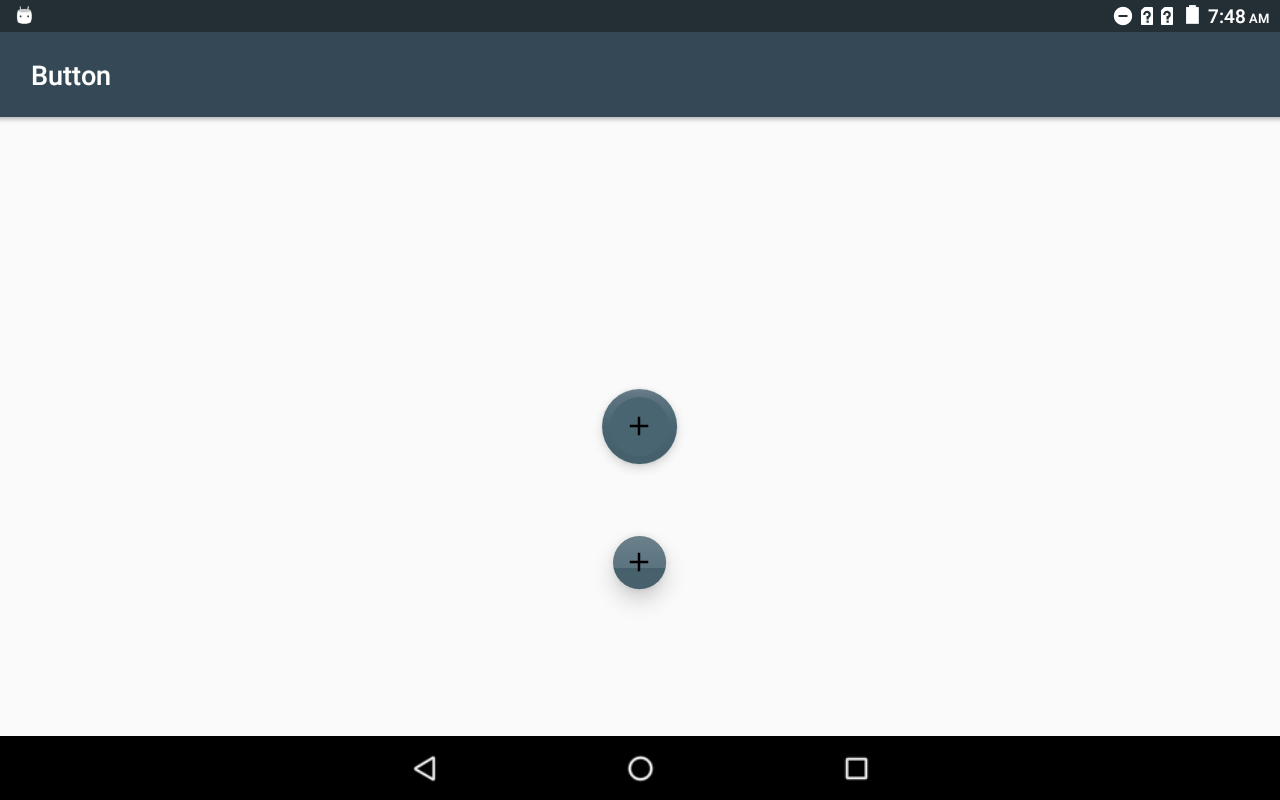
Text Button 1 is pressed.



Text Button is pressed.

**Widget.MaterialComponents.FloatingActionButton**

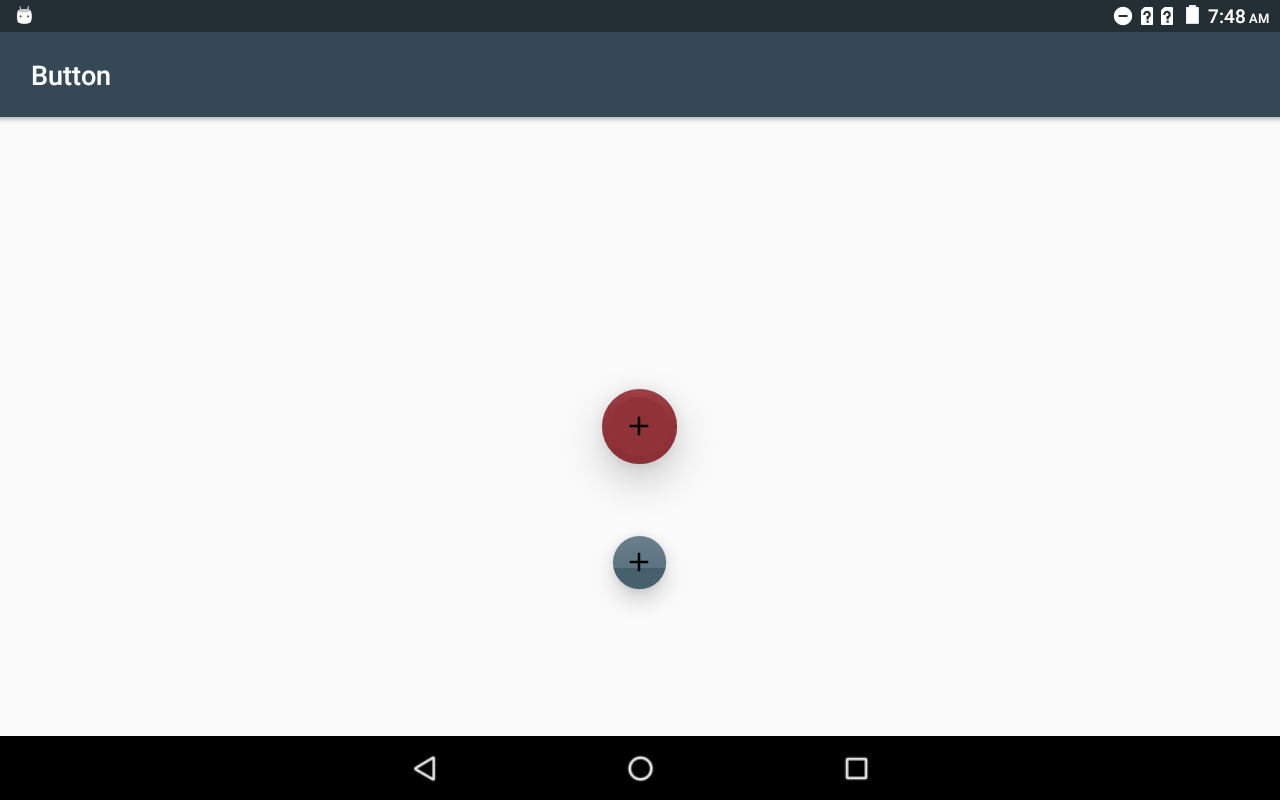
Its parent is Widget.Design.FloatingActionButton. It is a circular button that usually contains an icon in the middle. Its custom attributes are elevation, hoveredFocusedTranslationZ(in hovered state, it set the z axis as 2dp), pressedTranslationZ(in pressed state, it set the z axis as 6dp), rippleColor(same as the material button), showMotionSpec(there is a custom object animator class for that attribute) and hideMotionSpec(there is a custom object animator class for that attribute), fabSize(set the size of button, three options: mini, normal, auto) and fabCustomSize(custom your fab size besides the mini and normal).



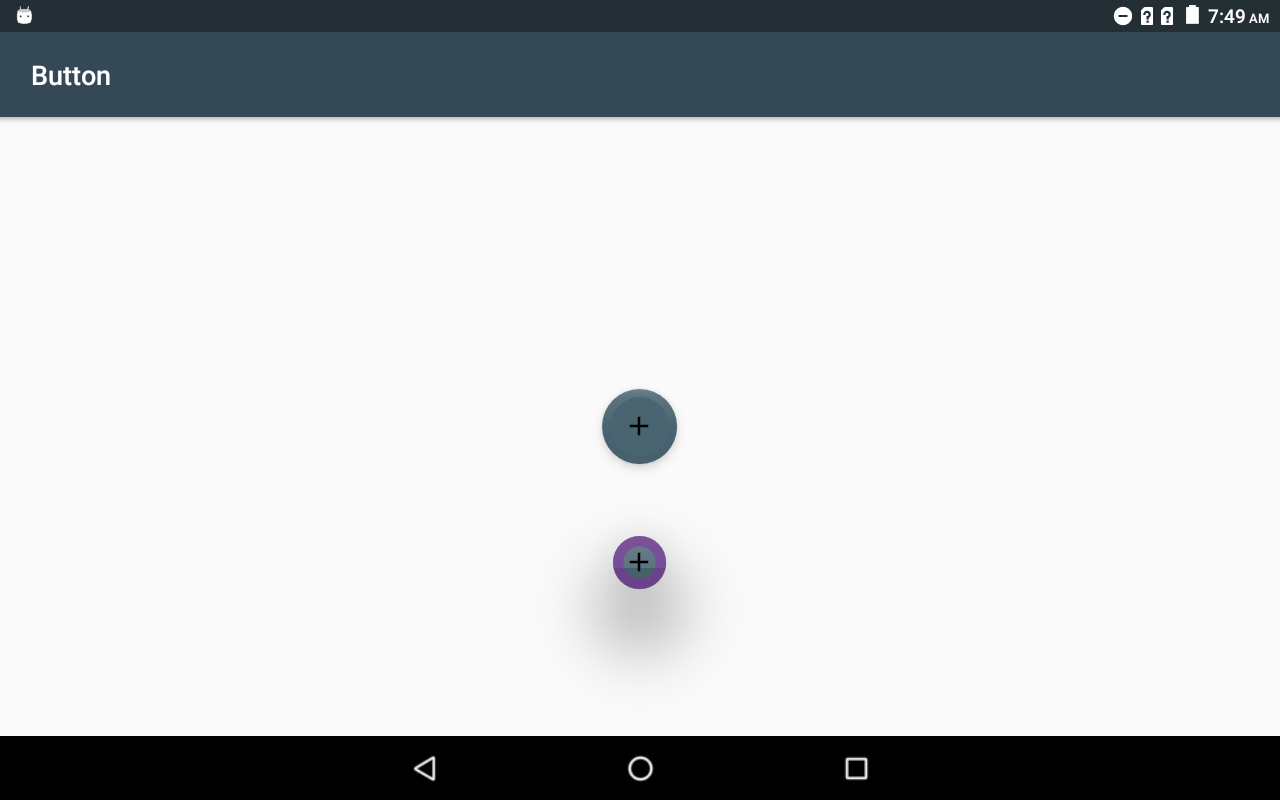
fabSize: The upper one is normal, bottom one is mini.

borderWidth: 6dp in upper one, 24dp in bottom one (default 0.5dp)

elevation is 12 dp in mini fab.



Upper fab is pressed. Ripple color is set as red. PressedTranslationZ is 16dp.



Mini fab is pressed. Ripple color is set as purple. PressedTranslationZ is 32dp. Middle does not turn purple because of the borderWidth value.

**References**

Homepage. (n.d.). Retrieved June 27, 2018, from <https://material.io/>

Android Developers. (n.d.). Retrieved June 27, 2018, from https://developer.android.com/