download material design android

Download material design android.

Completing the CAPTCHA proves you are a human and gives you temporary access to the web property.

What can I do to prevent this in the future?

If you are on a personal connection, like at home, you can run an anti-virus scan on your device to make sure it is not infected with malware.

If you are at an office or shared network, you can ask the network administrator to run a scan across the network looking for misconfigured or infected devices.

Another way to prevent getting this page in the future is to use Privacy Pass. You may need to download version 2.0 now from the Chrome Web Store.

Cloudflare Ray ID: 669d64f82a29c424 • Your IP : 188.246.226.140 • Performance & security by Cloudflare.

Download material design android.

TextInputLayout is a new element introduced in Design Support library to display the floating label in EditText. To display floating label in EditText, TextInputLayout needs to wrapped the EditText. We can also display the error message to EditText by using setError() and setErrorEnabled() methods. It takes the value of hint assigned to EditText and displays it as floating label. Android Design Support Library introduced some important new widgets that helps us to create consistent UI.

Floating Labels: Floating labels first introduced in Android design support library to display floating label over EditText. Firstly it acts as hint in the EditText when the field is empty. After that when a user start inputting the text it starts animating by moving to floating label position.

Special Note: In Android, one of the most basic UI element or widgets is an EditText. It is generally used to take the any kind of input from the user but what it lacks was a label attached to it. Therefore in most of the implementations hint was used as a label for the EditText. From the time material design was released, a new concept of floating labels was introduced. In this concept initially showed a label as a hint and when a user enters a value in the EditText that hint moves on to the top of the EditText as a floating label.

Table Of Contents.

Basic TextInputLayout XML Code:

Important Methods Of TextInputLayout:

Let’s we discuss some important methods of TextInputLayout that may be called in order to manage the TextInputLayout.

1. setError(CharSequenceerror): This method is used to set an error message that will be displayed below our EditText. In this method we set CharSequence value for error message.

Below we set the error message that will be displayed below our EditText.

2. getError(): This method is used for getting the error message that was set to be displayed using setError(CharSequence). It returns null if no error was set or error displaying is not enabled. This method returns CharSequence type value.

Below we firstly set the error and then get the error message that was set to be displayed using setError(CharSequence) method.

3. setErrorEnabled(boolean enabled): This method is used to set whether the error functionality is enabled or not in this layout. We set true for enabled and false for disabled error functionality.

Below we set the true value that enabled the error functionality.

4. isErrorEnabled(): This method is used to check whether the error functionality is enabled or not in this layout. This method returns Boolean type value which we set using setErrorEnabled (boolean enabled) method.

Below we firstly enabled the error functionality and then check whether the error functionality is enabled or not.

5. setHint(CharSequence hint): This method is used to set the hint to be displayed in the floating label, if enabled. In this method we set CharSequence value for displaying hint.

Below we set the hint to be displayed in the floating label.

6. getHint(): This method is used to get the hint which is displayed in the floating label, if enabled. This method returns CharSequence type value.

Below we set the hint and then get the hint which is displayed in the floating label.

7. setCounterMaxLength(int maxLength): This method is used to set the max length value to display at the character counter. In this method we pass int type value for setting max length value.

Below we set the max length value to display at the character counter.

8. getCounterMaxLength(): This method is used for getting the max length shown at the character counter. This method returns int type value which we set through setCounterMaxLength(int maxLength) method.

Below we firstly set the max length value and then get the max length value that shown at the character counter.

9. setCounterEnabled(boolean enabled): This method is used to set whether the character counter functionality is enabled or not in this layout. . We set true for enabled and false for disabled the counter functionality.

Below we set the true value that enabled the counter functionality in this layout.

10. setTypeface(Typeface typeface): This method is used to set the typeface to use for both the expanded and floating hint.

Below we set Sans – Serif typeface to use for the expanded and floating hint.

11. getTypeface(): This method is used for getting the typeface used for both the expanded and floating hint. This method returns Typeface type value which we set using setTypeface(Typeface typeface) method.

Below we firstly we set Sans – Serif typeface and then get the typeface used for both the expanded and floating hint.

Attributes of TextInputLayouts:

Now let's we discuss some common attributes of a TextInputLayout that helps us to configure it in our layout (xml).

1. support.design:counterMaxLength: This attribute is used to set the max length to display in the character counter. In this attribute we set int type value for setting max length value. We can also do this programmatically using setCounterMaxLength(int maxLength) method.

Below we set the max length value to display at the character counter.

2. support.design:counterEnabled: This attribute is used to set whether the character counter functionality is enabled or not in this layout. We set true for enabled and false for disabled the counter functionality. We can also do this programmatically using setCounterEnabled(boolean enabled) method.

Below we set the true value that enabled the counter functionality in this layout.

3. support.design:errorEnabled: This attribute is used to set whether the error functionality is enabled or not in this layout. We set true for enabled and false for disabled error functionality. We can also do this programmatically using setErrorEnabled(boolean enabled) method.

Below we set the true value that enabled the error functionality.

4. android:hint: This attribute is used to set the hint to be displayed in the floating label. We can also set hint programmatically using setHint(CharSequence hint) method.

Below we set the hint to be displayed in the floating label.

TextInputLayout/ Floating Labels In EditText Example In Android Studio:

Below is the example to show the usage of TextInputLayout where we create a login form with floating labels, input validations and error messages enabled. In this we also display a Sign In Button and perform click event on it so whenever a user click on Button we check the Fields and if a field is empty we display the error message otherwise we display "Thank You" message by using a Toast.

Step 1: Create a new project and name it TextInputLayoutExample.

Step 2: Open Gradle Scripts > build.gradle and add Design support library dependency.

Step 3: Open res -> layout -> activity_main.xml (or) main.xml and add following code:

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

In this step we open MainActivity and add the code for initiate the views (TextInputLayout and other views). After that we perform click event on Button so whenever a user click on Button we check the Fields and if a field is empty we display the error message otherwise we display "Thank You" message by using a Toast.

Output:

Now run the app and you will see login form on the screen. Click on email & password and you will see label text is Floating which looks beautiful.

TabLayout Tutorial With Example In Android Studio.

In Android TabLayout is a new element introduced in Design Support library. It provides horizontal layout to display tabs on the screen. We can display more screens in a single screen using tabs. We can quickly swipe between the tabs. TabLayout is basically view class required to be added into our layout(xml) for creating Sliding Tabs. We use different methods of TabLayout to create, add and manage the tabs.

Special Note: TabLayout is used to display tabs on the screen. We can create sliding as well as non sliding tabs by using TabLayout. If we need simple tabs without sliding then we replace the layout with the fragment on tab selected listener event and if we need sliding tabs then we use Viewpager.

Table Of Contents.

Basic TabLayout XML Code:

Important Methods Of TabLayout:

Let's we discuss some important methods of TabLayout that may be called in order to manage the TabLayout.

1. newTab(): This method is used to create and return a new TabLayout.Tab.

Below we create a new tab and set the text and icon for the tab.

2. addTab(Tab tab): This method is used to add a tab in the TabLayout. By using this method we add the tab which we created using newTab() method in the TabLayout. The tab will be added at the end of the list and If it is the first tab to be added then it will become the selected tab.

Below we firstly create a new tab and then add it in the TabLayout.

3. addTab(Tab tab, boolean setSelected): This method is used to add a tab in the TabLayout and set the state for the tab. By using this method we add the tab which we created using newTab() method in the TabLayout. In this method we also set the state of the tab whether it is selected or not.

Below we firstly create a new tab and then add it in the TabLayout and set the true value for setSelected parameter that makes it selectable.

4. addTab(Tab tab, int position): This method is used to add a tab in the TabLayout and set the state for the tab. By using this method we add the tab which we created using newTab() method in the TabLayout. The tab will be inserted at the given position. If it is the first tab to be added then it will become the selected tab.

Below we firstly create a new tab and then add it in the TabLayout at a specific position.

5. addTab(Tab tab, int position, boolean setSelected): This method is used to add a tab at a specific position and set the state of the tab. By using this method we add the tab which we created using newTab() method in the TabLayout. The tab will be inserted at the defined position and a Boolean value used to set the state of the tab. True value is used to make the tab selectable.

Below we firstly create a tab and then add it in the TabLayout at a specific position and we also set true value to make the tab selectable.

6. getSelectedTabPosition(): This method is used to get the position of the current selected tab. This method returns an int type value for the position of the selected tab. It returns -1 if there isn't a selected tab.

Below we get the current selected tab position.

7. getTabAt(int index): This method is used to get the tab at the specified index. This method returns TabLayout.Tab.

Below we get the tab at 1th index.

8. getTabCount(): This method is used to get the number of tabs currently registered with the action bar. This method returns an int type value for the number of total tabs.

Below we get the total number of tabs currently registered with the action bar.

9. setTabGravity(int gravity): This method is used to set the gravity to use when laying out the tabs.

Below we set the gravity for the tabs.

10. getTabGravity(): This method is used to get the current gravity used for laying out tabs. This method returns the gravity which we set using setTabGravity(int gravity) method.

Below we firstly set the gravity and then get the current gravity used for laying out tabs.

11. setTabMode(int mode): This method is used to set the behavior mode for the Tabs in this layout. The valid input options are:

MODE_FIXED: Fixed tabs display all tabs concurrently and are best used with content that benefits from quick pivots between tabs.

MODE_SCROLLABLE: Scrollable tabs display a subset of tabs at any given moment and it can contain longer tab labels and a larger number of

tabs. They are best used for browsing contexts in touch interfaces when users don't need to directly compare the tab labels. This mode is commonly used with a ViewPager.

Below we set the behaviour mode for the tabs.

12. getTabMode(): This method is used to get the current mode of TabLayout. This method returns an int type value which we set using setTabMode(int mode) method.

Below we firstly set the tab mode and then get the current mode of the TabLayout.

13. setTabTextColors(int normalColor, int selectedColor): This method is used to set the text colors for the different states (normal, selected) of the tabs.

Below we set the tab text colors for the both states of the tab.

14. getTabTextColors(): This method is used to get the text colors for the different states (normal, selected) of the tabs. This method returns the text color which we set using setTabTextColors(int normalColor, int selectedColor) method.

Below we firstly set the text colors and then get the text colors for the both states of the tab.

15. removeAllTabs(): This method is used to remove all tabs from the action bar and deselect the current tab.

Below we remove all the tabs from the action bar and deselect the current tab.

16. setOnTabSelectedListener(OnTabSelectedListener listener): This method is used to add a listener that will be invoked when tab selection changes.

Below we show how to use addOnTabSelectedListener of TabLayout.

17. removeTab(Tab tab): This method is used to remove a tab from the layout. In this method we pass the TabLayout.Tab object to remove the tab from the layout. If the removed tab was selected then it will be automatically deselected and another tab will be selected if present in the TabLayout.

Below we firstly create and add a tab and then remove it from the TabLayout.

18. removeTabAt(int position): This method is used to remove a tab from the layout. In this method we pass the position of the tab that we want to remove from the layout. If the removed tab was selected then it will be automatically deselected and another tab will be selected if present in the TabLayout.

Below we remove the tab from a specified position of TabLayout.

19. setSelectedTabIndicatorColor(int color): This method is used to set the tab indicator's color for the currently selected tab.

Below we set the red color for the selected tab indicator.

20. setSelectedTabIndicatorHeight(int height): This method is used to set the tab indicator's height for the currently selected tab.

Below we set the height for the selected tab's indicator.

21. setupWithViewPager(ViewPager viewPager): This method is used for setting up the TabLayout with ViewPager. ViewPager is mainly used for creating Sliding tabs.

Below we set the TabLayout with the ViewPager.

Attributes of TabLayout:

Now let's we discuss some common attributes of a TabLayout that helps us to configure it in our layout (xml).

1. id: id attribute is used to uniquely identify a TabLayout.

2. support.design:tabBackground: This attribute is used to set the background of the tabs. We can set a color or drawable in the background of tabs.

Below we set the Black color for the background of the tabs.

3. support.design:tabGravity: This attribute is used to set the gravity to use when laying out the tabs. We can also set gravity programmatically means in java class using setTabGravity(int gravity) method.

Below we set the gravity for the tabs.

4. support.design:tabIndicatorColor: This attribute is used to set the Color of the indicator used to show the currently selected tab. We can also set

color programmatically means in java class using setSelectedTabIndicatorColor(int color) method.

Below we set the red color for the selected tab indicator.

5. support.design:tabIndicatorHeight: This attribute is used to set the height of the indicator used to show the currently selected tab. We can also set height programmatically means in java class using setSelectedTabIndicatorHeight(int height) method.

Below we set the height for the selected tab's indicator.

6. support.design:tabMaxWidth: This attribute is used to set the maximum width for the tabs.

Below we set the maximum width value for the tabs.

7. support.design:tabMinWidth: This attribute is used to set the minimum width for the tabs.

Below we set the minimum width value for the tabs.

8. support.design:tabMode: This attribute is used to set the behavior mode for the Tabs in this layout. We can also set the mode programmatically means in java class using setTabMode(int mode) method.

Below we set the behaviour mode for the tabs.

9. support.design:tabPadding: This attribute is used to set the padding along all edges of tabs.

android.support.design:tabPaddingBottom: Set padding along the bottom edge of the tabs.

android.support.design:tabPaddingEnd: Set padding along the end edge of the tabs.

android.support.design:tabPaddingStart: Set padding along the start edge of the tabs.

android.support.design:tabPaddingTop: Set padding along the top edge of the tabs.

Below we set the padding along all edges of the tabs.

10. support.design:tabSelectedTextColor: This attribute is used to set the text color to be applied to the currently selected tab. We can also set this programmatically using setTabTextColors(int normalColor, int selectedColor) method.

Below we set the text color for the selected tab.

11. support.design:tabTextColor: This method is used to set the default text color to be applied to tabs. . We can also set this programmatically using setTabTextColors(int normalColor, int selectedColor) method.

Below we set the default text color for the tabs.

TabLayout Example In Android Studio:

Example 1 of TabLayout:

Below is the first example of TabLayout in which we display three non-sliding tabs. In this example we define a TabLayout and a FrameLayout in our xml file. In this example we create and add 3 tabs in the TabLayout with the help of different methods of TabLayout. After that we implement setOnTabSelectedListener event and replace the FrameLayout with the fragment's according to selected tab.

Step 1: Create a new project and name it TabLayoutExample.

Step 2: Open build.gradle and add Design support library dependency.

Step 3: Open res -> layout ->activity_main.xml (or) main.xml and add following code:

In this step we add the code for displaying TabLayout and ViewPager in our xml file.

Step4: Open src -> package -> MainActivity.java.

In this step we open MainActivity and add the code for initiate the FrameLayout and TabLayout. After that we create and add 3 tabs in the TabLayout. Finally we implement setOnTabSelectedListener event and replace the FrameLayout with the fragment's according to selected tab.

Step 5: Now we need 3 fragments and 3 xml layouts for three tabs. So create three fragments by right click on your package folder and create classes and name them as FirstFragment, SecondFragment and ThirdFragment and add the following code respectively.

FirstFragment.class.

SecondFragment.class.

ThirdFragment.class.

Step 6: Now create 3 xml layouts by right clicking on res/layout -> New -> Layout Resource File and name them as fragment_first, fragment_second and fragment_third and add the following code in respective files.

Here we will design the basic simple UI for all the three tabs.

fragment_first.xml.

fragment_second.xml.

fragment_third.xml.

Example 2 of TabLayout Using ViewPager:

Below is the example of TabLayout in which we display sliding tabs with the help of ViewPager. In this example we define a TabLayout and a ViewPager in our xml file. In this example we create and add 3 tabs in the TabLayout with the help of different methods of TabLayout. After that we create a PagerAdapter to set up the TabLayout with ViewPager.

Step 1: Create a new project and name it TabLayoutExample.

Step 2: Open build.gradle and add Design support library dependency.

Step 3: Open res -> layout ->activity_main.xml (or) main.xml and add following code:

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

In this step we open MainActivity and add the code for initiate the ViewPager and TabLayout. After that we create and add 3 tabs in the TabLayout. Finally we set an Adapter named PagerAdapter to set up the TabLayout with ViewPager.

Step 5: Create a new Class named PagerAdapter and extend FragmentStatePagerAdapter in the class. In this step we create a PagerAdapter class and extends FragmentStatePagerAdapter In the class for setting the TabLayout with the ViewPager.

Step 6: Now we need 3 fragments and 3 xml layouts for three tabs. So create three fragments by right click on your package folder and create classes and name them as FirstFragment, SecondFragment and ThirdFragment and add the following code respectively.

FirstFragment.class.

SecondFragment.class.

ThirdFragment.class.

Step 7: Now create 3 xml layouts by right clicking on res/layout -> New -> Layout Resource File and name them as fragment_first, fragment_second and fragment_third and add the following code in respective files.

Download material design android.

Material Design is a visual language developed by Google which was first introduced with Lollipop OS and since then it has become popular in designing and developing Android Apps. As per Google this language is based on paper and ink.

Here our in-depth tutorial will teach how to design beautiful and user friendly Application using Android Material Design. All our tutorial will have at least 1 practical example and step by step explanation of each topic.

Prequisites For Leaning Android Material Design:

JAVA For Android:

JAVA is a programming language which is most commonly used in Android App Development. Before you start learning Material Design you will need to learn Object Oriented Java. Check out our JAVA for Android section to learn it.

Android UI Basics:

You will also need to be familiar with Android UI basics.

Android Studio:

Android Studio is the official IDE (integrated development environment) for developing Android Apps by Google. It is available for free download on Windows, Mac OS X and Linux.

Check out our Android Studio guide to get started with it.

Android Material Design Tutorials:

Below are the list of Material design topics. Follow the link to read full tutorial:

Download material design android.

PercentRelativeLayout in Android is a subclass of RelativeLayout that supports percentage based margin and dimensions for Views(Button, TextView or any other view).

Percent Support Library: Percent Support Library provides a feature to set the dimensions and margins in the term of Percentage. It has two pre built Layouts- the PercentRelativeLayout and PercentFrameLayout. In this article we focused on PercentRelativeLayout. This library is pretty easy to use because it has same Relative Layout and FrameLayout that we are familiar with, just with some additional new functionalities.

Important Note – To use percentrelativelayout you need to add Percent Support Library dependency in build.gradle file.

Gradle Scripts > build.gradle (Module:App) -> inside dependencies.

Table Of Contents.

PercentRelativeLayout Vs Relative Layout In Android:

PercentRelativeLayout class extends from Relative Layout class so it supports all the features, attributes of RelativeLayout and it has percentage dimensions so it also supports some features of LinearLayout. In Simple words we can say that PercentRelativeLayout has features of both Layouts with reduced view complexity.

Need of PercentRelativeLayout In Android.

In Android there are lot of Layout's that can be used at the time of development but at last we always end up with our main three layouts Linear Layout, Relative Layout and FrameLayout. In case if we need to create complex view then we use weight property of LinearLayout to distribute view across screens. But while using weight property we must have noticed that we have to add a default container view to encapsulate our child view . We can follow this approach but it adds an additional view hierarchy in our layout's which is of no use except holding weight sum value and child view. After the introduction of PercentRelativeLayout we can put percentage dimension and margins to our view that helps us to remove the problem of existing RelativeLayout.

Short Description About RelativeLayout:

Relative Layout is very flexible layout used in android for custom layout designing. It gives us the flexibility to position our component's based on the relative or sibling component's position. Just because it allows us to position the component anywhere we want so it is considered as most flexible layout. For the same reason Relative layout is the most used layout after the Linear Layout in Android. It allow its child view to position relative to each other or relative to the container or another container. You can read full tutorial about relative layout.

Important Note: In PercentRelativeLayout, it is not necessary to specify layout_width and layout_height attributes if we specify layout_widthPercent attribute. If in case we want the view to be able to take up more space than what percentage value permits then we can add layout_width and layout_height attributes with wrap_content value. In that case if the size of content is larger than percentage size then it will be automatically resized using wrap_content rule. If we don't use layout_height and layout_width attribute then android studio shows us an error marker on element in the editor but it will be automatically ignored at run-time.

Basic PercentRelativeLayout XML Code:

Attributes of PercentRelativeLayout.

Now let's we discuss some common attributes of a PercentRelativeLayout that helps us to configure it in our layout (xml). It supports all the attributes of RelativeLayout but here we are only discuss some additional attributes.

Very Important Note: Before you use below code make sure you have added Percent Support Library dependency in build.gradle file in depencies.

Gradle Scripts > build.gradle (Module:App) -> inside dependencies.

layout_widthPercent: This attribute is used to set the width of the view in percentage. Below we set 50% value for the widthPercent and wrap_content value for the height of the TextView.

PercentRelativeLayout Example In Android Studio:

Below is the example of PercentRelativeLayout in which we create a Login Form with 2 fields user name and Password. In this example we take 2 EditText, 2 TextView and 1 Login Button. For these views we set the dimensions and margin in form of percentage. If a user click on the login Button then a "Thank You" message with user name is displayed on the screen with the help of Toast.

Step 1: Create A New Project And Name It PercentRelativeLayoutExample.

Step 2: Open Gradle Scripts > build.gradle and add Percent Support Library dependency.

Step 3: Open res -> layout -> activity_main.xml (or) main.xml and add following code: In this Step we take 2 EditText's, 2 TextView's and 1 Login Button and for these views we set the dimensions and margin in form of percentage. Note that in all the view's i didn't take layout_widht attribute and it still works fine. Android Studio shows a error line on element but it will be automatically ignored at runtime.

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

In this step we open the MainActivity and add the code for initiates the views(EditText and Button). After that we implement setOnClickListener event on login Button so that whenever a user click on Button a thank you message if a user already fill both fields or a error message that enter your user name and password is displayed on the screen with the help of Toast.

Output:

Now run the App and you will login form designed using PercentRelativeLayout.