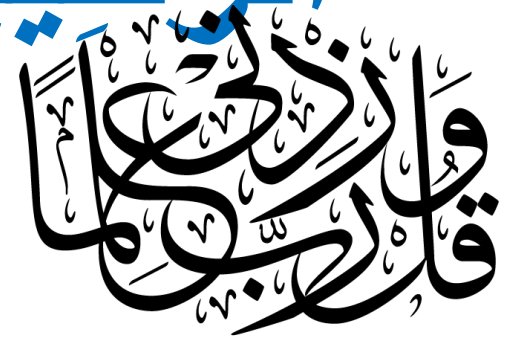
# بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ


قَالُواْ سُبۡحَٰنَكَ
لَا عِلۡمَ لَنَآ إِلَّا مَا عَلَّمۡتَنَآۖ
إِنَّكَ أَنتَ ٱلۡعَلِيمُ ٱلۡحَكِيمُ




# Dialog, Animations

# Mobile Computing

## LECTURE 13

Dialog, Animations

Instructor:

Maulana Haq Nawaz

# تصحیح نیت

حضرت محمد صلی اللہ علیہ وسلم نے فرمایا

اِنَّمَا الْاَعْمَالُ بِالنِّیَّاتِ

ترجمہ

اعمال کا دارومدار نیتوں پر

# Little Efforts Daily Will Make You the Greatest

> To systematically learn and get excellence in any concept / subject

- روز کا کام روز کریں
- اک مہینے کا کھانا ایک دن میں نہیں کھایا جا سکتا،
ایسے ہی ایک مہینے کا کام ایک دن میں نہیں ہو سکتا

# Little Efforts Daily Will Make You the

## Importance of completing tasks on daily basis

### Main Reason of Failure in Life

- یہ کام کل کریں گے
- جو کام کبھی بھی ہو سکتا ہے وہ کبھی نہیں ہوتا
- زندگی ایک دن ہے اور وہ ہے آج۔ زندگی میں کل نام کی کوئی چیز نہیں ہے
- جو دن آپ کی زندگی سے چلا گیا اب واپس نہیں آئے گا
- آج کا کام آج ہی ہوسکتا ہے
- جو گز گیا وہ آنا نہیں ، آنے والے دن کا پتہ نہیں ، آج میدان جما ہے تو اپنے جوہر دکھاؤ

# How to Achieve BIG Goals in Life

**Balanced Life is Ideal Life ?**

**To achieve BIG Goals in Life**

- Make a Schedule of 24 Hours with a focus on Five main components of Human Life
  - Health
    - Physical Health
    - Mental Health
    - Social Health
  - Spirituality
  - Work
  - Family
  - Friends

# جو کام کریں دل سے کریں

کام کرنا.

خوشی خوشی کام کرنا.

اللہ کو ساتھ لے کرخوشی خوشی کام کرنا.

آیت: إِيَّاكَ نَعْبُدُ وإِيَّاكَ نَسْتَعِينُ

ترجمہ: یا اللہ ہم تیری ہی عبادت کرتے ہیں.
اور تجھ ہی سے مد د مانگتے ہیں

# Lecture Outline

1. **Revision**
2. **Dialog Overview**
3. **Alert Dialog**
4. **Alert Dialog with Multiple Choices**
5. **Customize Alerts**
6. **Stop Complaining! Stop Criticizing! Let's Start Contributing**
7. **Lecture Summary**

# Perfection and Neglection

قَدْ أَفْلَحَ الْمُؤْمِنُونَ الَّذِينَ هُمْ فِي صَلَاتِهِمْ خَاشِعُونَ (1-2 المؤمنون)

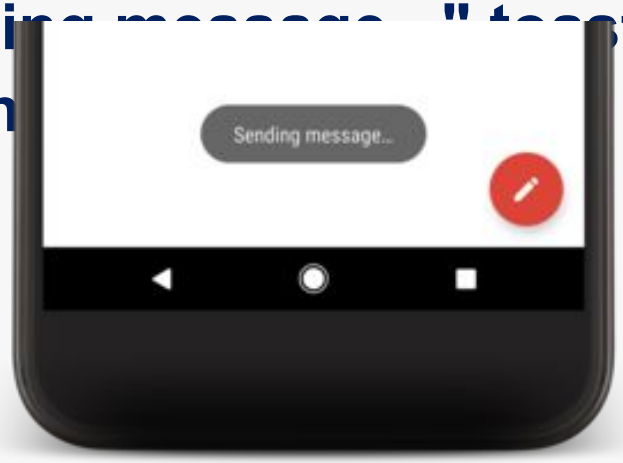
Success is really attained by the believers who concentrate their attention in humbleness when offering Salāh (prayers)

فَوَيْلٌ لِلْمُصَلِّينَ الَّذِينَ هُمْ عَنْ صَلَاتِهِمْ سَاهُونَ (4-5 الماعون)

So, Woe to those performers of Salāh, who are neglectful of their Salāh, {Expression of sorrow on some event}

# Toast

- A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.
- For example, clicking Send on an email triggers a "Sending message..." toast, as shown in the following screen




Toasts are not clickable. If user response to a status message is required, consider instead using a Notification.

# How to make Toast

- First, instantiate a Toast object with one of the makeText() methods. This method takes three parameters: the application Context, the text message, and the duration for the toast. It returns a properly initialized Toast object. You can display the toast notification with show(), as shown in the following example:

- Toast toast = Toast.makeText(this, "Toast Text", Toast.LENGTH_LONG);

  toast.show();

# Creating a Custom Toast

- If a simple text message isn't enough, you can create a customized layout for your toast notification. To create a custom layout, define a View layout, and pass the root View object to the setView(View) method.

# Creating a Custom Toast

20201018IntentNotification > app > src > main > res > layout > activity_main.xml

Android

activity_main.xml | MainActivity.java

```
1   package org.haqnawaz.a20201018intentnotification;
2
3   import ...
10
11  public class MainActivity ex
```

app
- manifests
- java
  - org
    - haqnaw
      - a202
        - M
  - org (andro
  - org (test)
- generatedJava
- res
  - drawable
  - layout
    - activity
  - mipmap
  - values
- Gradle Scripts

New
- Link C++ Project with Gradle
- Cut ⌘X
- Copy ⌘C
- Copy Path ⇧⌘C
- Copy Relative Path ⌥⇧⌘C
- Paste ⌘V
- Jump to Source ⌘↓
- Find Usages ⌥F7
- Analyze
- Refactor
- Validate
- Add to Favorites
- Reformat Code ⌥⌘L
- Optimize Imports ^⌥O
- Delete... ⌦
- Mark as Plain Text
- Reveal in Finder
- Open in Terminal
- Open in Browser

Kotlin File/Class
- Layout resource file
- Sample Data Directory
- File
- Scratch File ⇧⌘N
- Directory
- C++ Class
- C/C++ Source File
- C/C++ Header File
- Image Asset
- Vector Asset
- module-info.java
- .editorconfig file
- Kotlin Script
- Singleton
- Gradle Kotlin DSL Build Script
- Gradle Kotlin DSL Settings
- Edit File Templates...
- AIDL
- Activity
- Android Auto

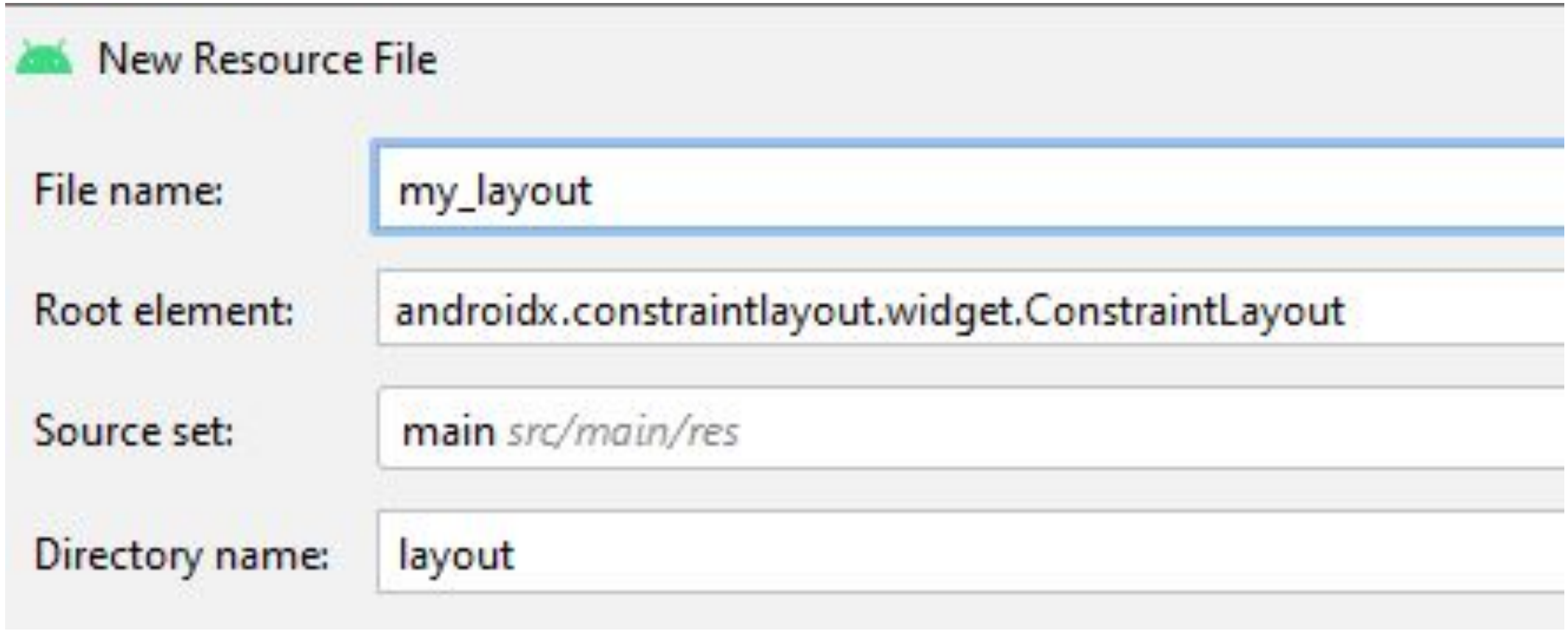
```
xOffset: 200,  yOffset: 0);
```

Build: | Build Output

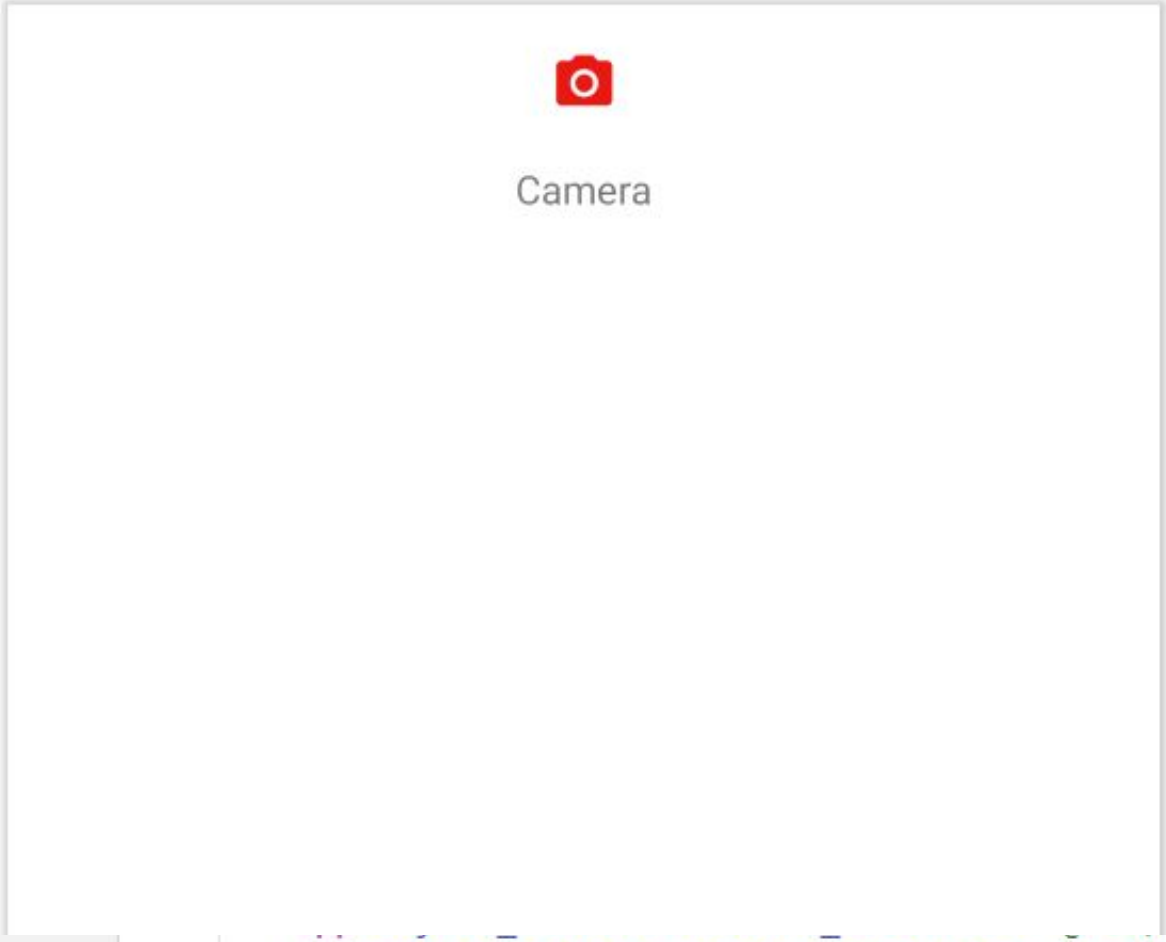
- Build: comp
  - Run buil
    - Load

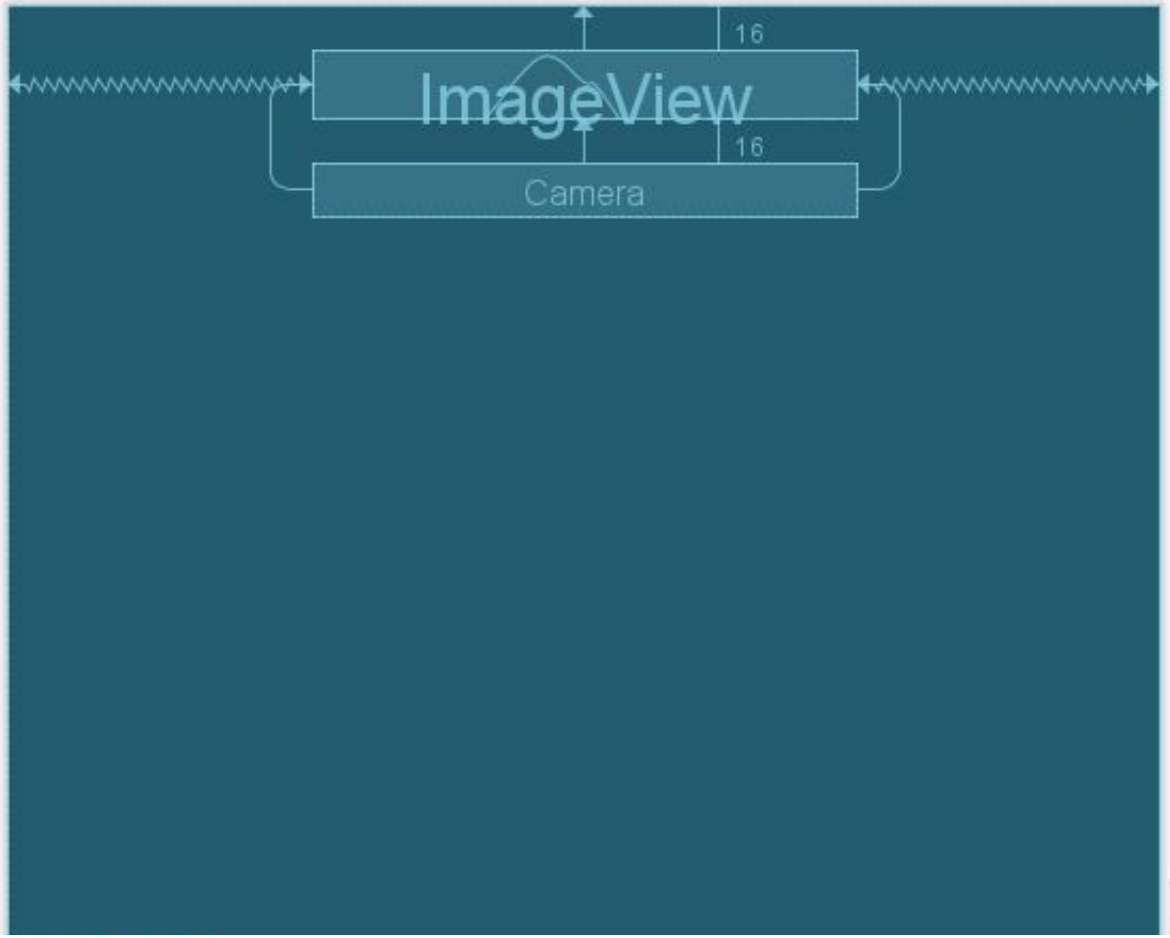
201018IntentNotification

# Creating a Custom Toast



# Creating a Custom Toast



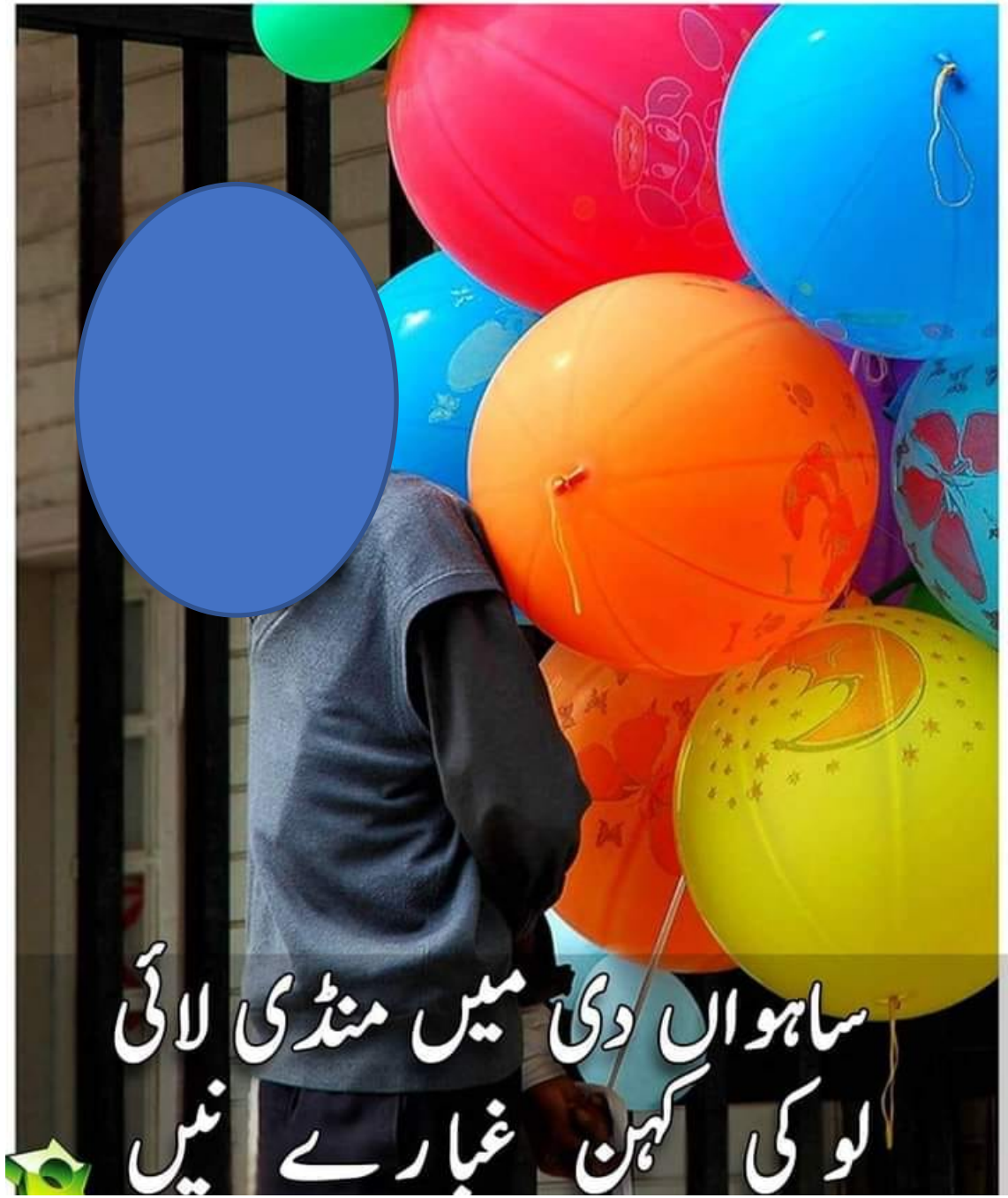

```
28        app:layout_constraintTop_toBottomOf="@+id/imageViewCamera" />
29    </androidx.constraintlayout.widget.ConstraintLayout>
```

# Code of Custom Toast

```
LayoutInflater layoutInflater = getLayoutInflater();
View myLayout=layoutInflater.inflate(R.layout.cust_toast,null);
ImageView imageView=myLayout.findViewById(R.id.imageViewCustToast);
imageView.setImageResource(R.drawable.a5);
TextView myMessage= myLayout.findViewById(R.id.textViewCustToast);
myMessage.setText("My Custom Toast");
Toast myToast=new Toast(getApplicationContext());
myToast.setDuration(Toast.LENGTH_LONG);
myToast.setView(myLayout);
myToast.show();
```

میری سانسیں خریدو گے ___

مجھے کچھ اور جینا ہے ___

# LayoutInflater

Instantiates a layout XML file into its corresponding View objects. It is never used directly. Instead, use Activity.getLayoutInflater() or Context#getSystemService to retrieve a standard LayoutInflater instance that is already hooked up to the current context and correctly configured for the device you are running on.

# Context

- Interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.
- The context class is an interface to global information about an application environment.
- Ways of accessing other parts/features of the program.

# Context what can be done

- Loading resources.
  - Color, image, string, sound, asset.
- Launching a new activity.
  - Intent requires parameter of context
- Creating views.
- Obtaining system service.
  - Camera, GPS etc

# Context example of

- **getResources().getColor()**
  - **Allows your app to read values from the resource file colors.xml**

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary"> #0d466b</color>
    <color name="colorPrimaryDark"> #0d466b</color>
    <color name="colorAccent"> #0d466b</color>
    <color name="appColor"> #0d466b</color>
    <color name="appLightColor"> #0d466b</color>
    <color name="appLightGreyText">#484848</color>
```

```java
    String[] webAppURL = getResources().getStringArray(R.array.webApps_URL);
    String[] webAppTitle = getResources().getStringArray(R.array.webApps_Title);
    String[] webApp_ArabicTitle = getResources().getStringArray(R.array.webApps_Arabic_Title);
    int[] webApp_ArabicIcon = getResources().getIntArray(R.array.webApps_Arabic_Icon);
```

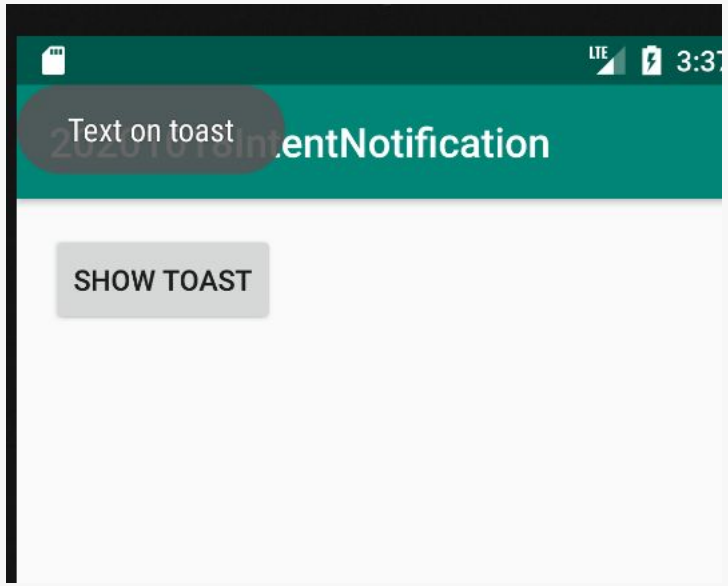
```xml
    <color name="translation_text_color">#40e3e4f9</color>
    <color name="tafseer_text_color">#73e3e4f9</color>

    <color name="trans_grey">#85A6AAAF</color>
    <color name="trans_dark_grey">#852d323b</color>
</resources>
```

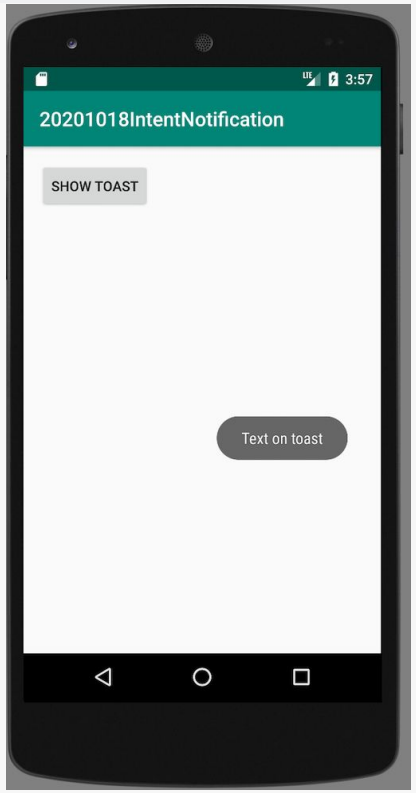
# getApplicationContext()

- Return the context of the single, global Application object of the current process. This generally should only be used if you need a Context whose lifecycle is separate from the current context, that is tied to the lifetime of the process rather than the current component.

# Toast Positioning

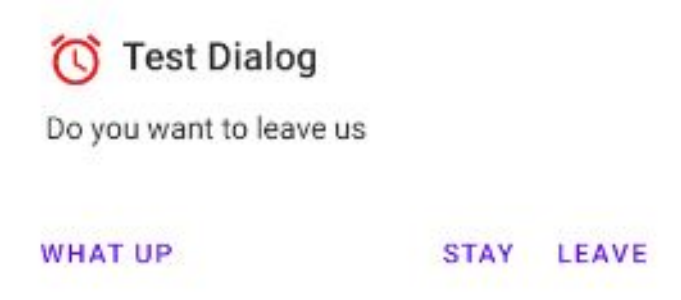
```
toast.setGravity( gravity: Gravity.CENTER|Gravity.RIGHT,  xOffset: 100,  yOffset: 200);
```



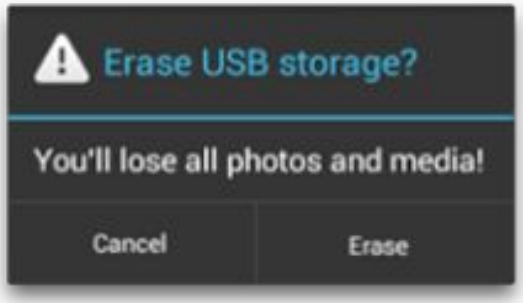



# Dialog

- A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.







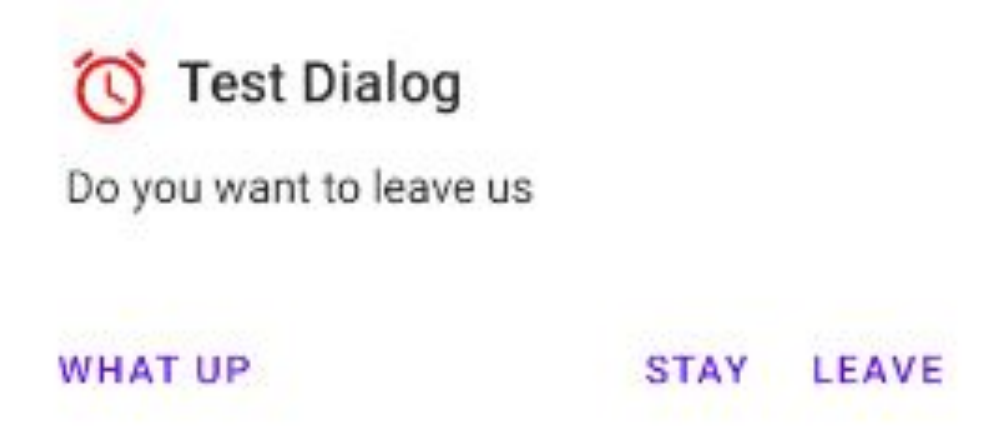
# Dialog

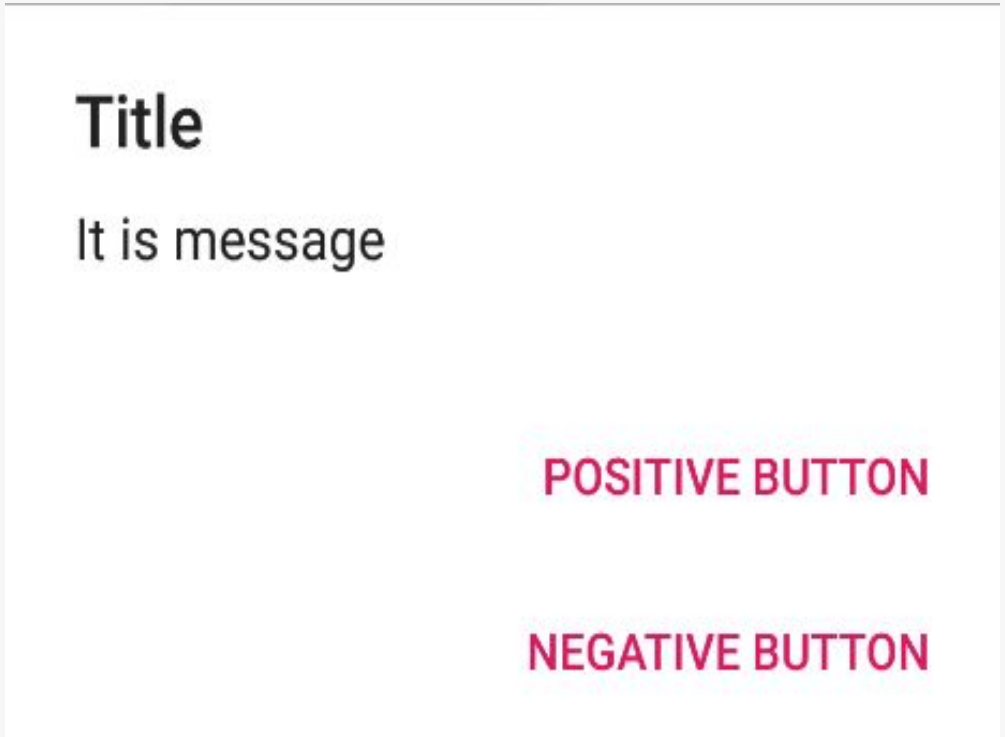
- The Dialog class is the base class for dialogs, but you should avoid instantiating Dialog directly. Instead, use one of the following subclasses:
  - AlertDialog A dialog that can show a title, up to three buttons, a list of selectable items, or a custom layout.
  - DatePickerDialog or TimePickerDialog A dialog with a pre-defined UI that allows the user to select a date or time.

# DialogFragment

- These classes define the style and structure for your dialog, but you should use a DialogFragment as a container for your dialog. The DialogFragment class provides all the controls you need to create your dialog and manage its appearance, instead of calling methods on the Dialog object
- Using DialogFragment to manage the dialog ensures that it correctly handles lifecycle events such as when the user presses the Back button or rotates the screen. The DialogFragment class also allows you to reuse the dialog's UI as an embeddable component in a larger UI, just like a traditional Fragment (such as when you want the dialog UI to appear differently on large and small screens).

# AlertDialog



# AlertDialog

```
// Create the object of AlertDialog Builder class
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

// Set the message show
builder.setMessage(”Message you want to show");
// Set Alert Title
builder.setTitle("Alert !");

// Set Cancelable false for when the user clicks on the outside the Dialog Box
then it will remain show
builder.setCancelable(false);
```

# AlertDialog

```
// Create the object of AlertDialog Builder class
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

// Set the message show
builder.setMessage(”Message you want to show");
// Set Alert Title
builder.setTitle("Alert !");

// Set Cancelable false for when the user clicks on the outside the Dialog Box
then it will remain show
builder.setCancelable(false);
```

# AlertDialog

```
// Set the positive button with yes name OnClickListener method is use of
DialogInterface interface.
builder.setPositiveButton(
        "Kill",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int asdf)
            {
                // When the user click yes button then app will close
                finish();
            }
        });
```

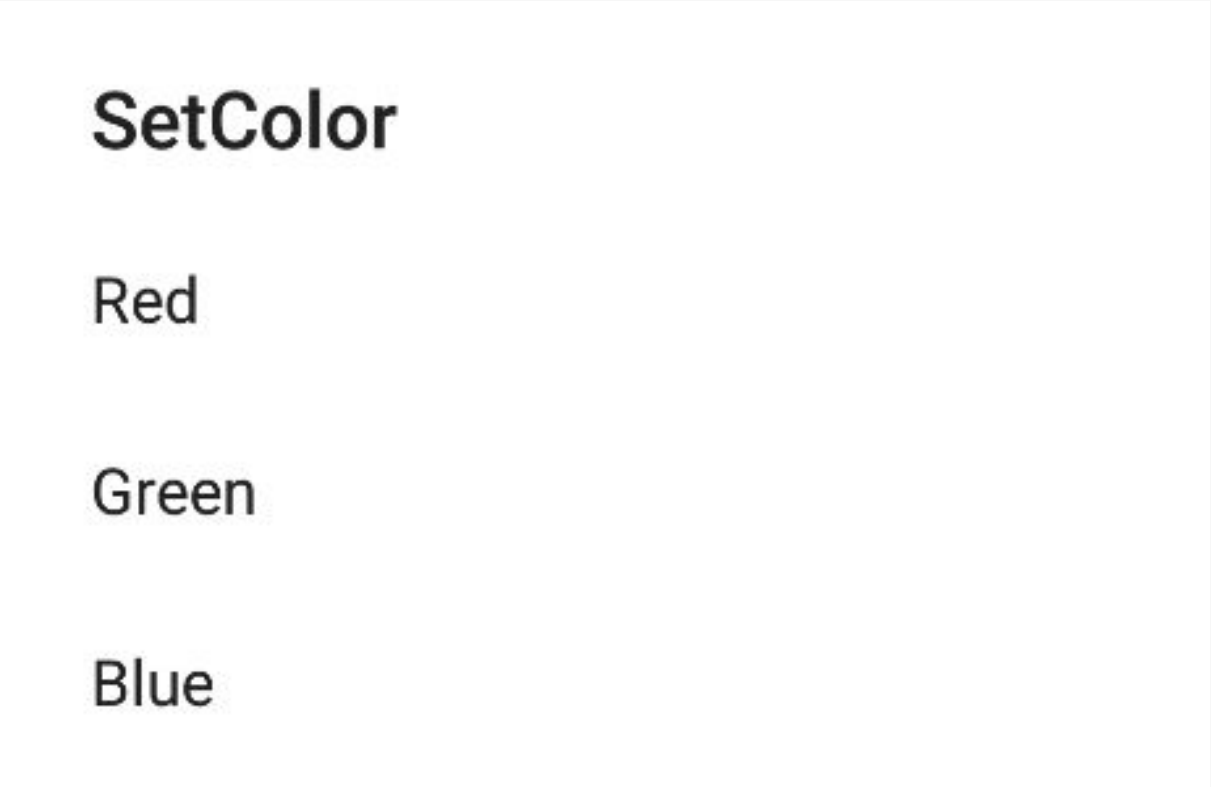
# AlertDialog

```
public void showDialog(View view) {
    AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
    builder.setMessage("It is message");
    builder.setTitle("Title");
    builder.setCancelable(false);
    builder.setPositiveButton(
        "Positive Button",
        new DialogInterface.OnClickListener() {
          @Override
          public void onClick(DialogInterface dialog, int asdf)
          {
            finish();
          }
        });
    builder.setNegativeButton(
        "Negative Button",
        new DialogInterface.OnClickListener() {

          @Override
          public void onClick(DialogInterface dialog, int which)
          {
            dialog.cancel();
          }
        });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
  }
}
```

# AlertDialog

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
.setIcon(R.drawable.ic_baseline_alarm_24)
                .setTitle("Test Dialog")
                .setMessage("Do you want to leave us")
                .setPositiveButton("Leave", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                    }
                })
                .setNegativeButton("Stay", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();
                    }
                })
                .setNeutralButton("What up", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(getApplicationContext(), "Click Leave to close and Stay to cancel",
Toast.LENGTH_LONG).show();
                    }
                }).show();
```

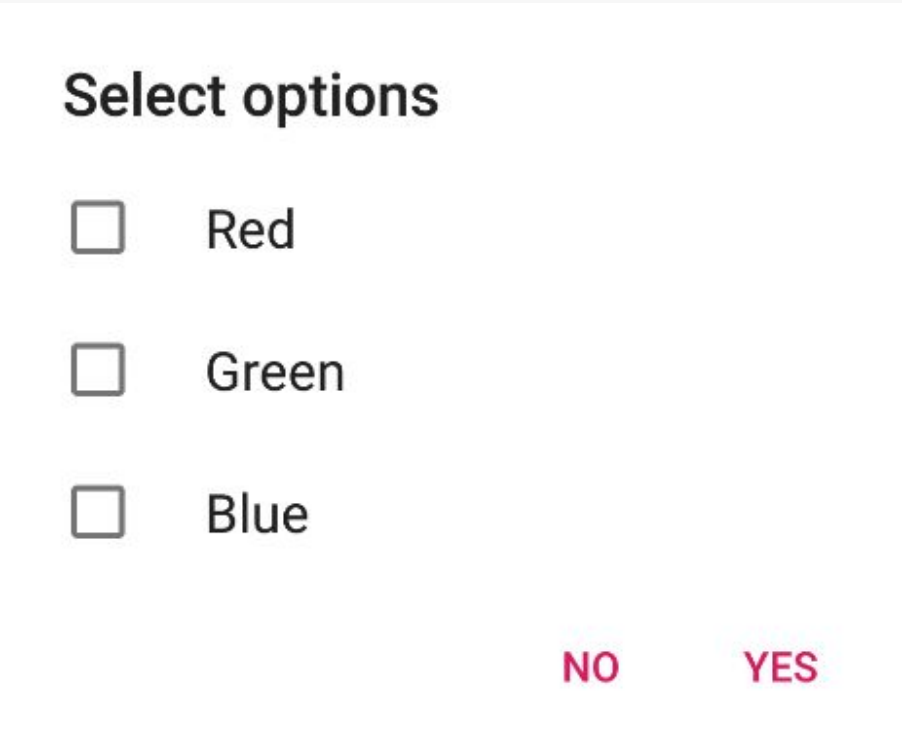
# AlertDialog with List

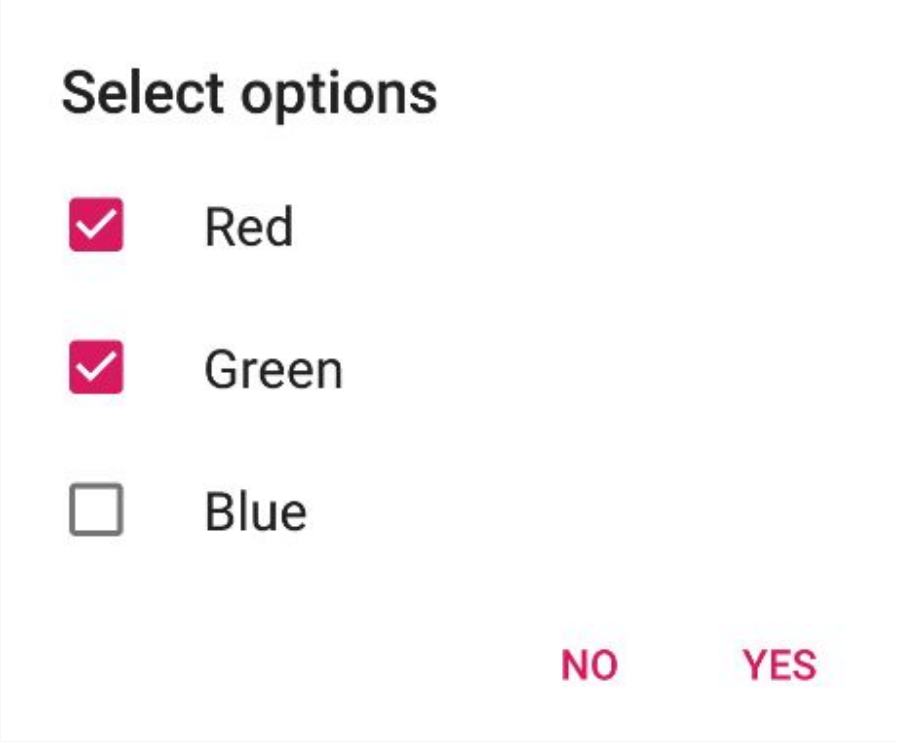


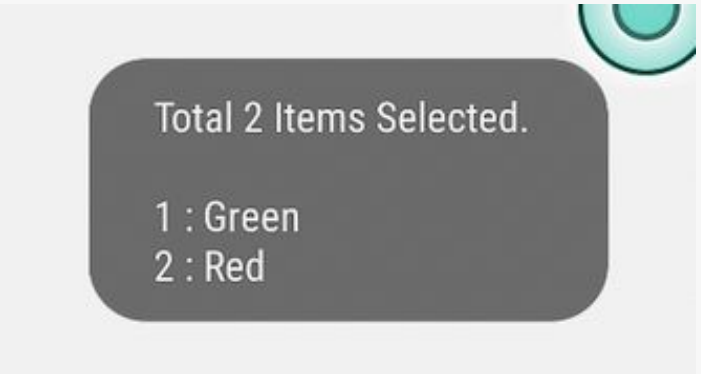
# AlertDialog with List

```
public class AlertDialogListActivity extends AppCompatActivity {
    String [] Colors = {"Red", "Green", "Blue"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_alert_dialog_list);

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("SetColor")
                .setItems(Colors, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(AlertDialogListActivity.this, Colors[which], Toast.LENGTH_SHORT).show();
                    }
                });
        AlertDialog dialog = builder.create();
        dialog.show();
    }
}
```

# AlertDialog with Multiple Choice



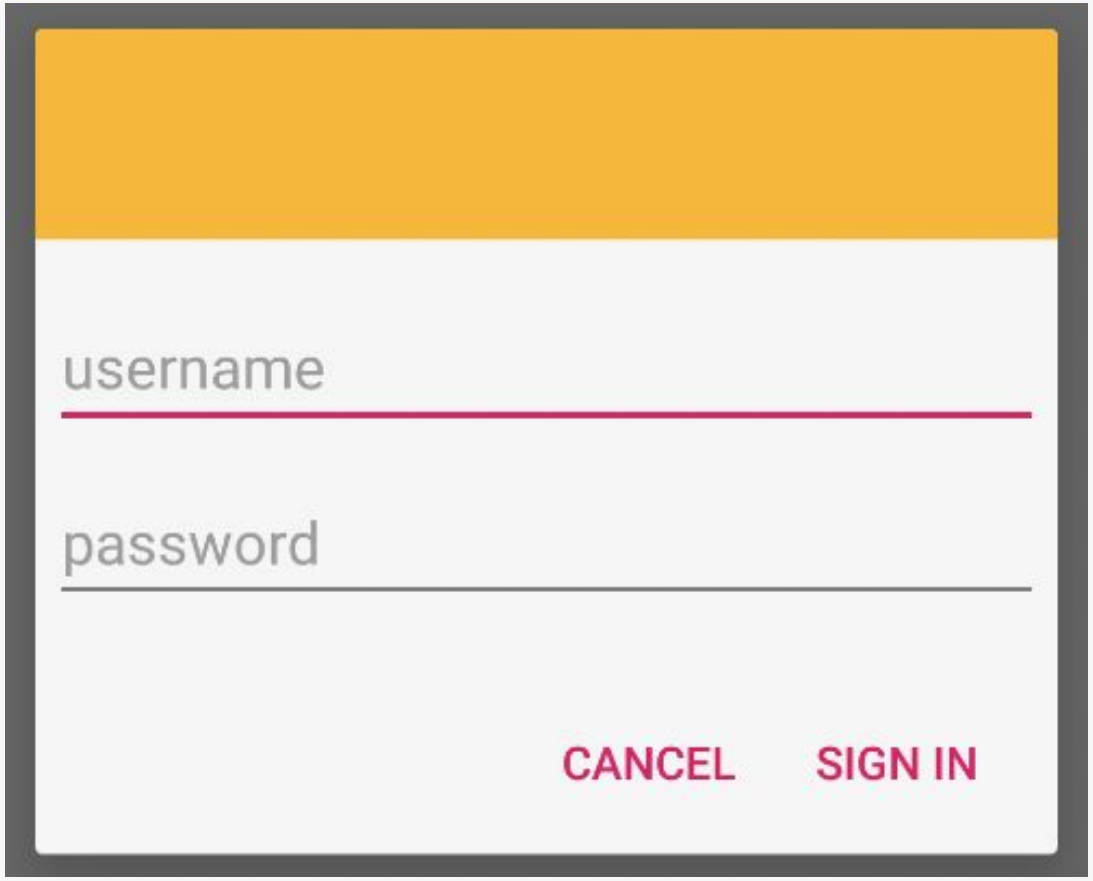
# AlertDialog with List

```
String [] Colors = {"Red", "Green", "Blue"};
   ArrayList<Integer> selectedItems = new ArrayList();  // Where we track the selected items
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_multiple_choice_dialog);
      AlertDialog.Builder builder = new AlertDialog.Builder(this);
      builder.setTitle("Select options")
            .setMultiChoiceItems(Colors, null,
                  new DialogInterface.OnMultiChoiceClickListener() {
                     @Override
                     public void onClick(DialogInterface dialog, int which, boolean isChecked) {
                        if (isChecked) {
                           selectedItems.add(which);
                        } else if (selectedItems.contains(which)) {
                           selectedItems.remove(Integer.valueOf(which));
                        }
                     }
                  })
            .setCancelable(false)
```

# AlertDialog with List

```
            .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    String msg = "";
                    for (int i = 0; i < selectedItems.size(); i++) {
                        msg = msg + "\n" + (i + 1) + " : " + Colors[ selectedItems.get(i)];
                    }
                    Toast.makeText(getApplicationContext(), "Total " + selectedItems.size() + " Items Selected.\n" + msg,
Toast.LENGTH_SHORT).show();
                }
            })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Toast.makeText(MultipleChoiceDialogActivity.this,"No Option Selected",Toast.LENGTH_SHORT).show();
                }
            });
        AlertDialog dialog  = builder.create();
        dialog.show();
    }
}
```

# AlertDialog with Customize Layout



# AlertDialog with List

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

  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  LayoutInflater inflater = this.getLayoutInflater();

  builder.setView(inflater.inflate(R.layout.sign_in, null))
      // Add action buttons
          .setPositiveButton("Sign In", new DialogInterface.OnClickListener() {
        @Override
            public void onClick(DialogInterface dialog, int id) {
          // sign in the user ...
            }
      })
      .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
        }
      });
  AlertDialog dialog  = builder.create();
```