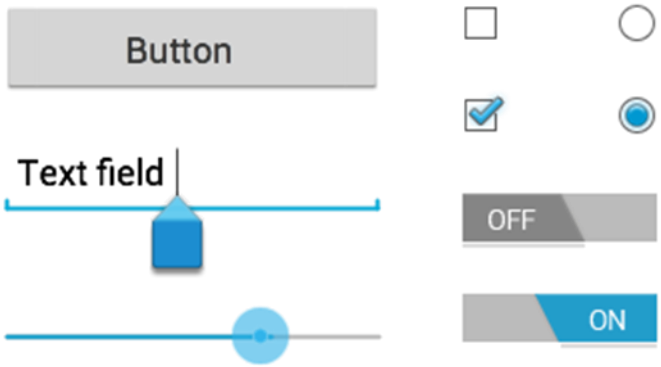
2.1 User Interface

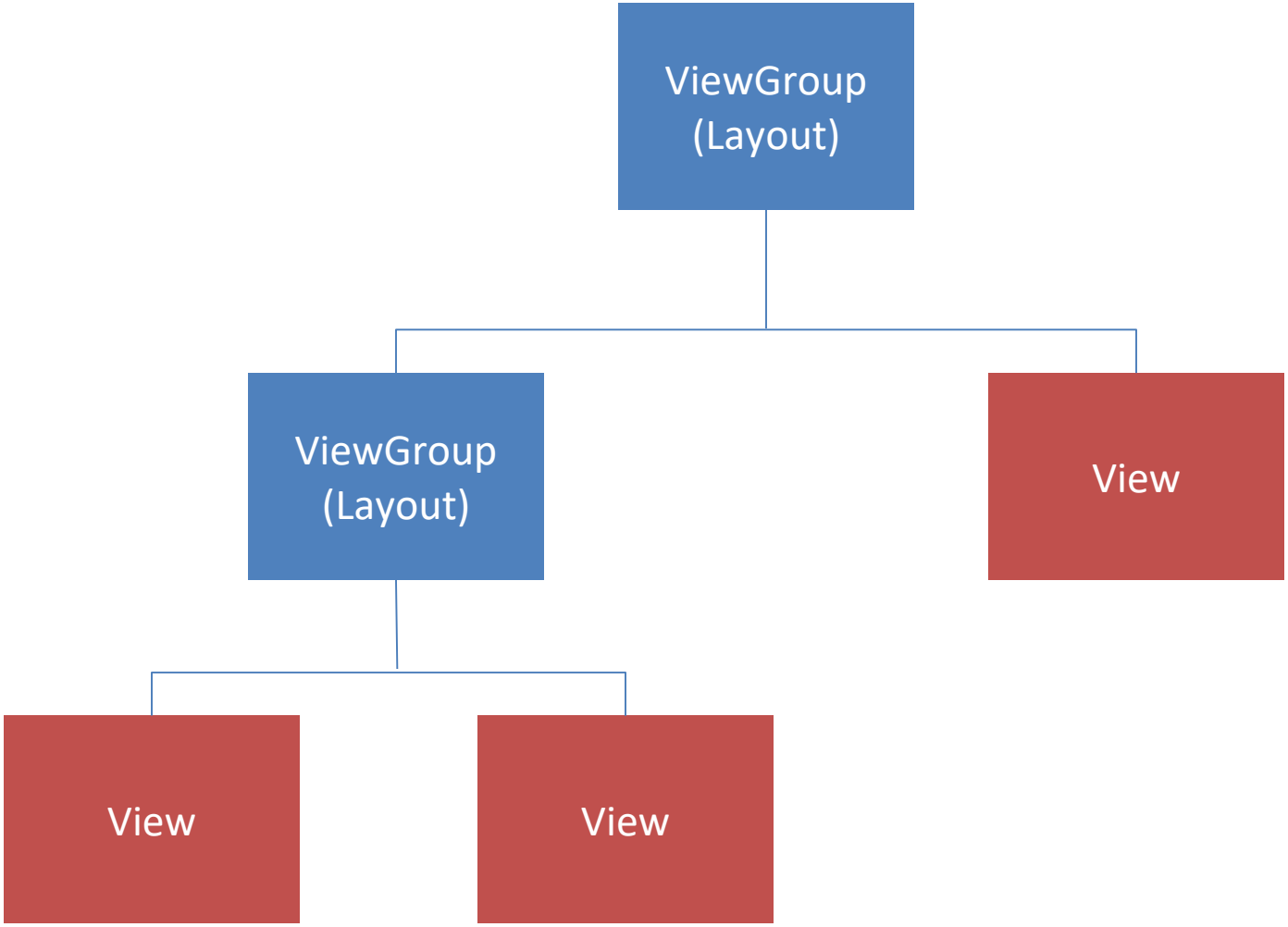
Layouts, Components and Input Events



Objectives

- Explain the overview of User Interface
- Create UI containing common UI components
- Create and use style and theme

Overview of User Interface (UI)



ViewGroup

- Usually called Layout
- You can nest one or more layouts within another layout
- Two ways to create Layout:
  - Declare UI elements in XML
  - Instantiate layout elements at runtime

ViewGroup

- Loading layout XML

Kotlin

```
fun onCreate(savedInstanceState: Bundle) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main_layout)
}
```

Loading UI

Java

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

ViewGroup – Static Content

1. Linear Layout

Organizes its children into a single horizontal or vertical row.

Creates a scrollbar if the length of the window exceeds the length of the screen.

2. Relative Layout

Enables you to specify the location of child objects relative to each other

*Has been replaced by the Constraint Layout

3. Web Layout

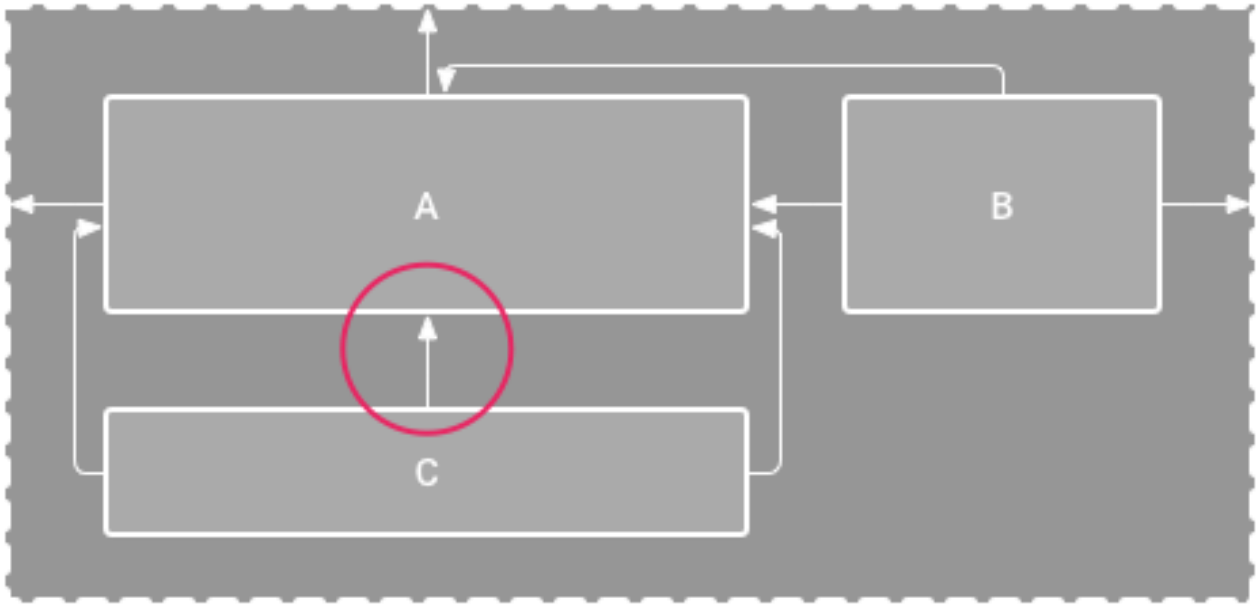
Displays web pages.

Constraint Layout

- Similar to Relative Layout
- Creates large and complex layouts with a flat view hierarchy
- Compatible with Android 2.3 (API level 9) and higher

Constraint Layout

- You must add at least one horizontal and one vertical constraint for the view

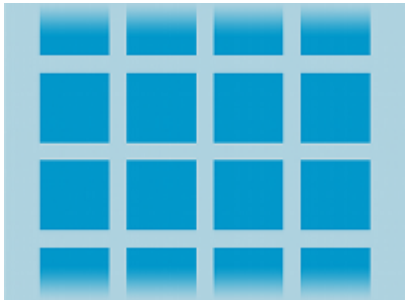


ViewGroup – Dynamic Content

4. List View

Displays a scrolling single column list.

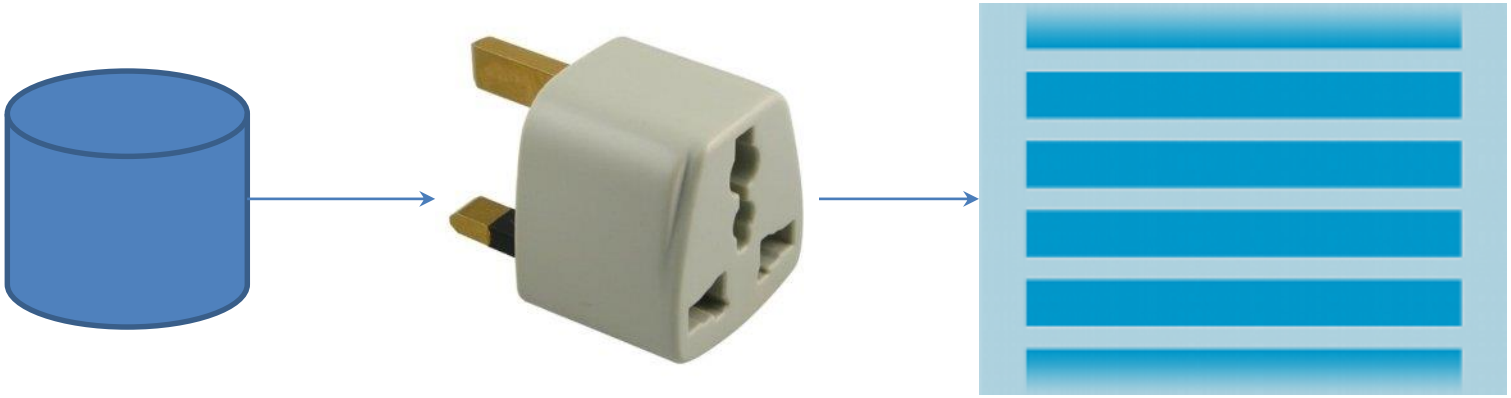
5. Grid View



Displays a scrolling grid of columns and rows

Adapter for ListView and GridView

- Suitable for content that is dynamic but small
- Uses an Adapter to bind data to a layout



Kotlin

```
val adapter = ArrayAdapter<String>(this,
                                   android.R.layout.simple_list_item_1,
                                   myStringArray)

val listView: ListView = findViewById(R.id.listview)

listView.adapter = adapter
```

Java

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1,
        myStringArray);

ListView listView = (ListView) findViewById(R.id.listview);

listView.setAdapter(adapter);
```

Note: The `android.R` refers to an Android system resource.

Kotlin

```
listView.onItemClickListener = AdapterView.OnItemClickListener { parent, view,
position, id ->
    // Do something in response to the click
}
```

Java

```
// Create a message handling object as an anonymous class.
private OnItemClickListener messageClickedHandler = new OnItemClickListener() {
    public void onItemClick(AdapterView parent, View v, int position, long id) {
        // Do something in response to the click
    }
};

listView.setOnItemClickListener(messageClickedHandler);
```

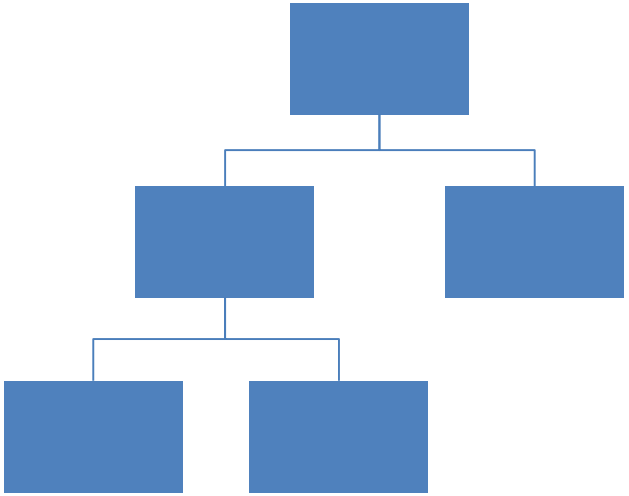
Improving Layout Performance

1. Optimizing Layout hierarchies
2. Re-using Layouts with <include> tag
3. Loading Views on demand

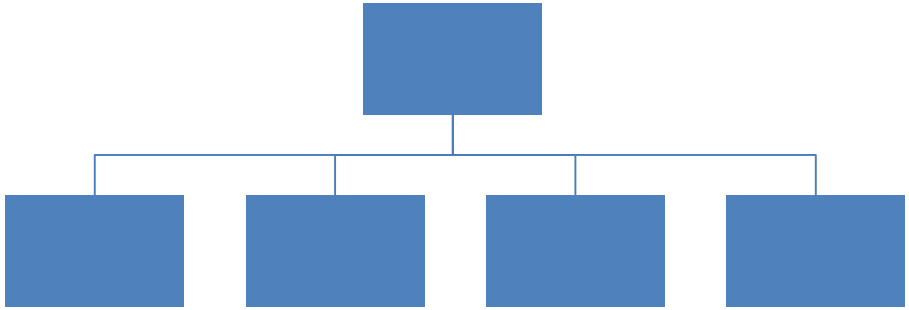
Question?

Which layout is better in term of performance?

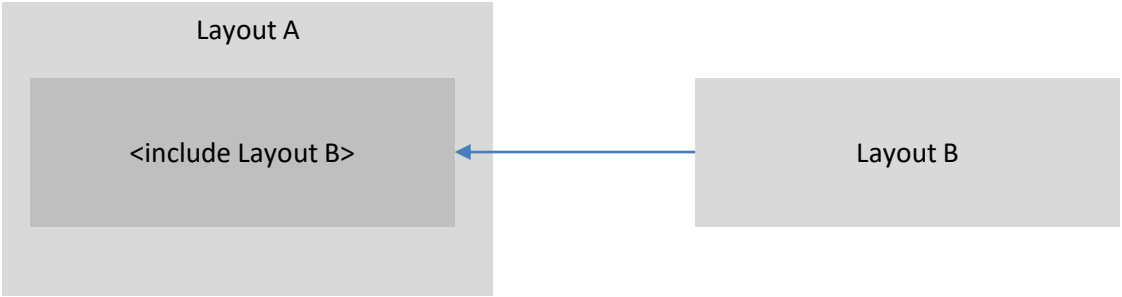
A



B



Using the <include> tag

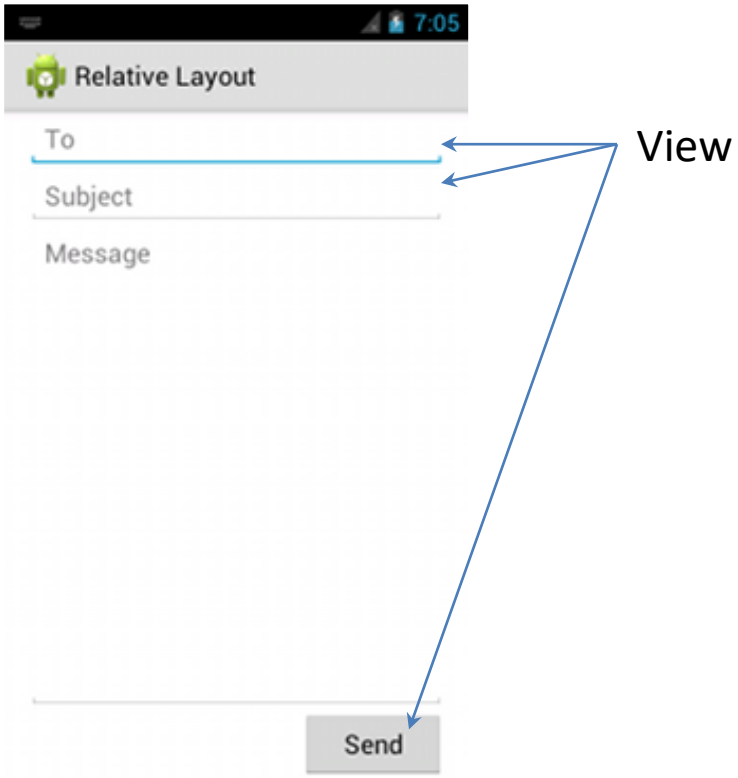


Layout B uses the `<merge>` element as the root view.

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

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/add"/>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/delete"/>

</merge>
```

View

- Also called widget
- E.g. Button and TextView
- Android uses XML to define View



View

- Each View has an ID, to uniquely identify it within UI

```
<TextView android:id="@+id/textMessage"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="“Your message" />

<Button android:id="@+id/buttonMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="“Show Message" />
```

View

- An instance of View object is created to link program code to UI, usually in the `onCreate()` method
- The `findViewByID` method is used to form the link

Kotlin

```
val myButton: Button = findViewById(R.id.buttonMessage)
```

Java

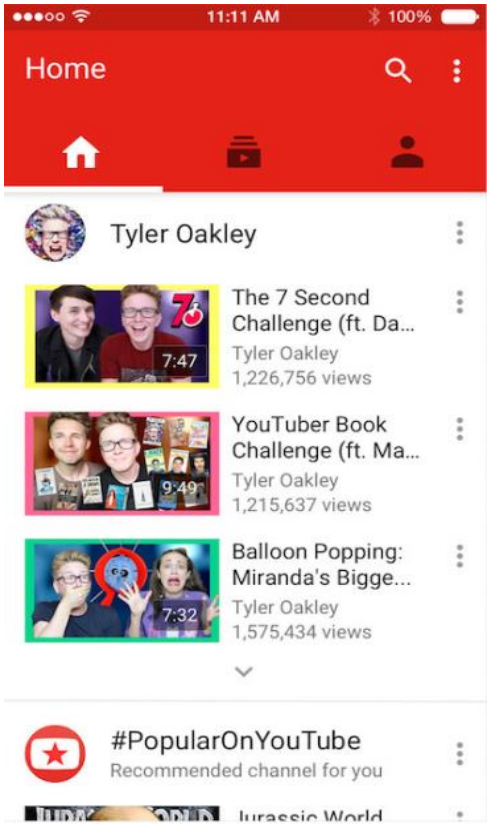
```
Button myButton = (Button) findViewById(R.id.buttonMessage);
```

The `R` is a special class that maintains a reference ID for each resource added to the project “res” folder

Question?

- Identify a suitable Layout for each of the following apps:

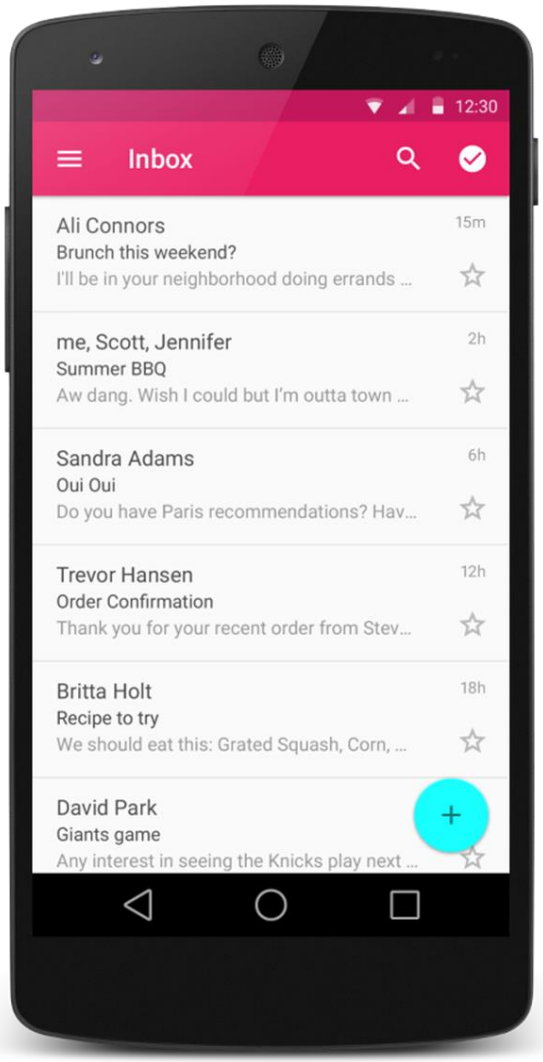
A



B

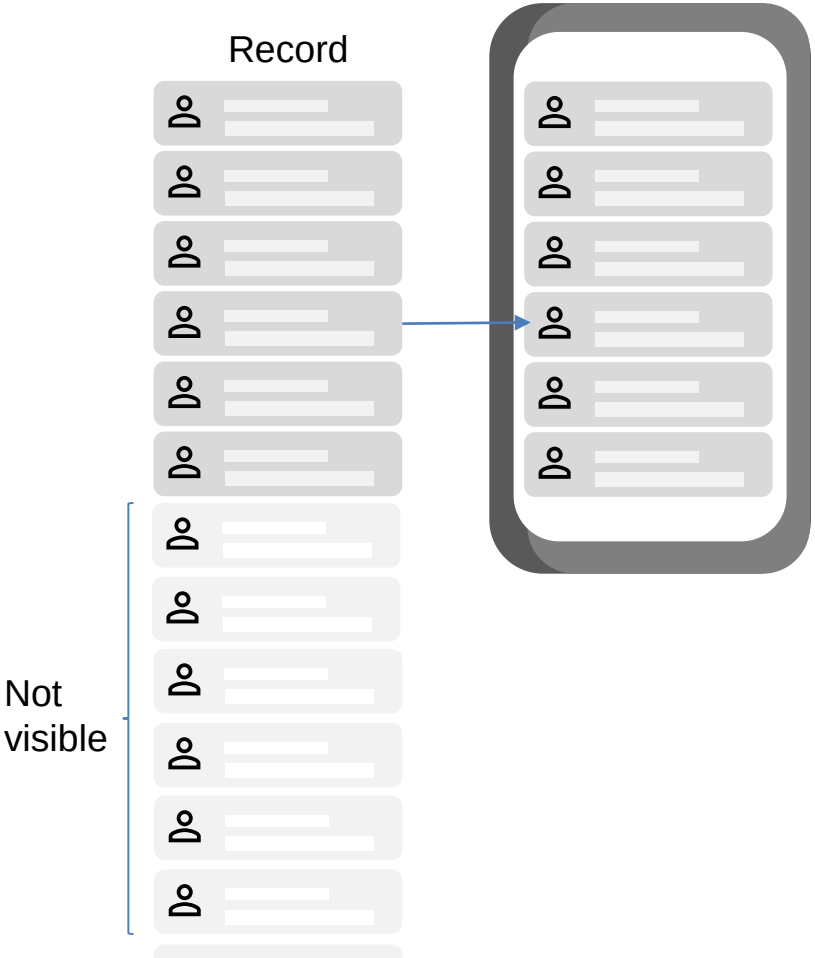
Recycler View

- It process a list of elements based on large data sets
- It displays items that are currently visible on the screen



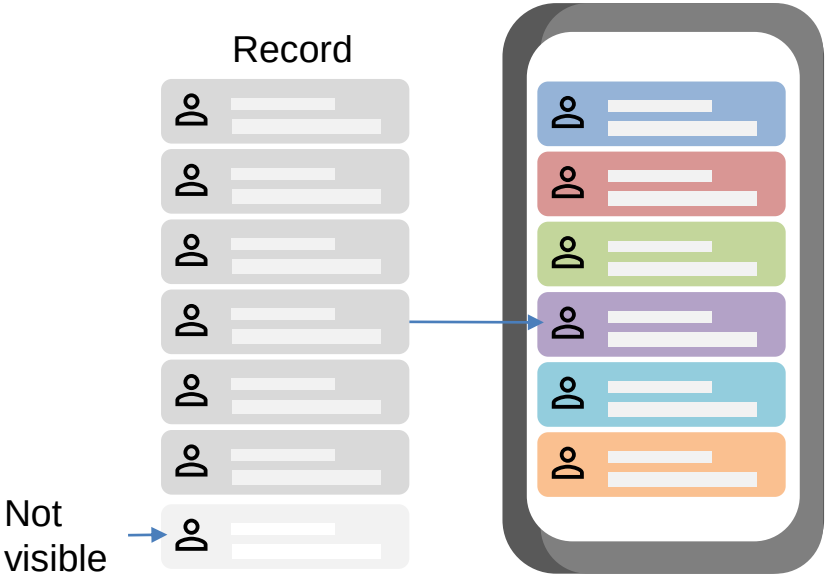
ListView

All records are loaded to the ListView



RecyclerView

Load records that are visible to the user using ViewHolder

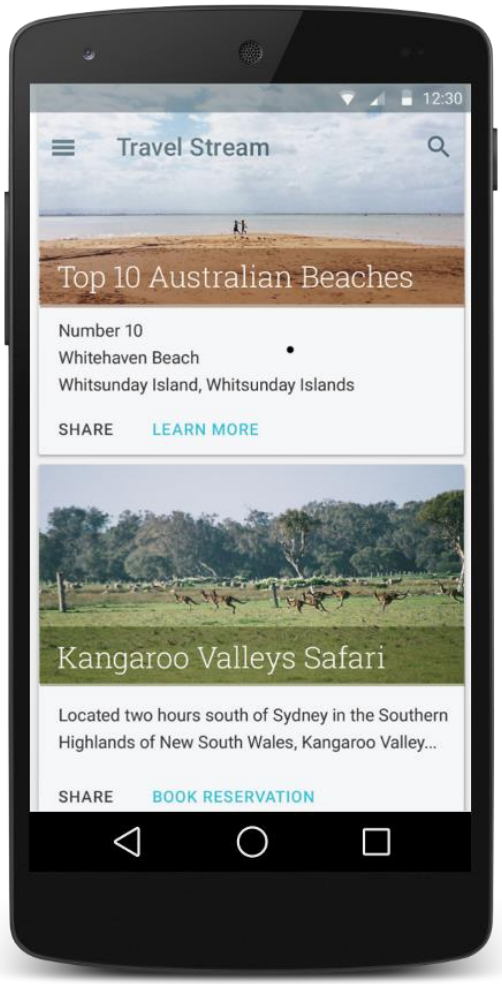


Recycler View

When an item scrolls off the screen, the item's views are recycled. That means the item is filled with new content that scrolls onto the screen.

Card-based Layout

- Allows app to show information inside cards
- Provides a consistent look across different platform



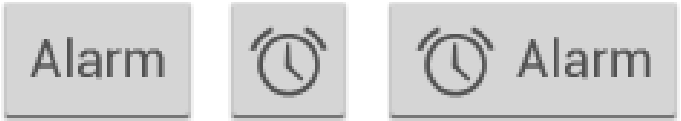
UI Components

1. Button
2. Text Field
3. Check Box
4. Radio Button
5. Toggle
6. Spinner

Button

- Two types:
  - Button class : text and an incom
  - ImageButton class: icon

- Two ways to handle click event:
  - use android.onClick attribute in layout
  - use setOnClickListener() method in code



https://developer.android.com/guide/topics/ui/controls/button

Button – Click Events

```
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Kotlin

```
/** Called when the user touches the button */
fun sendMessage(view: View) {
    // Do something in response to button click
}
```

Java

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Button – Click Events

Kotlin

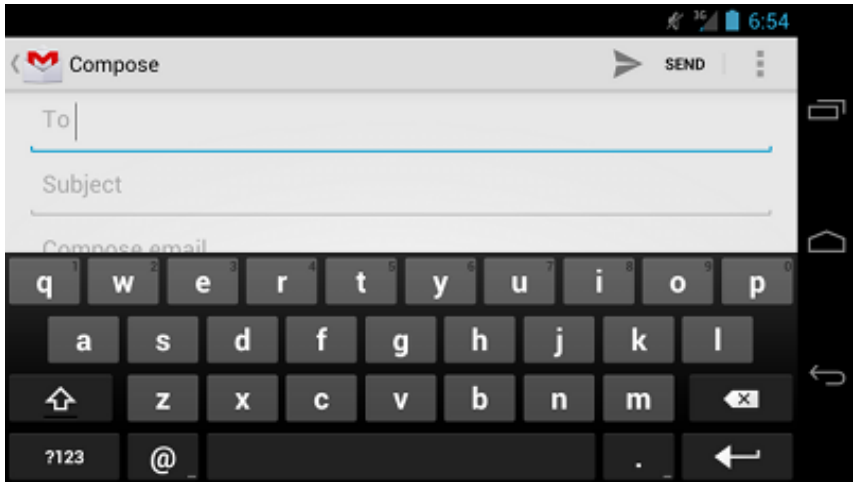
```
val button: Button = findViewById(R.id.button_send)
button.setOnClickListener {
    // Do something in response to button click
}
```

Java

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

Text Fields

- You may specify the keyboard type in the layout with the android:inputType attribute




```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

https://developer.android.com/training/keyboard-input/style

Text Fields

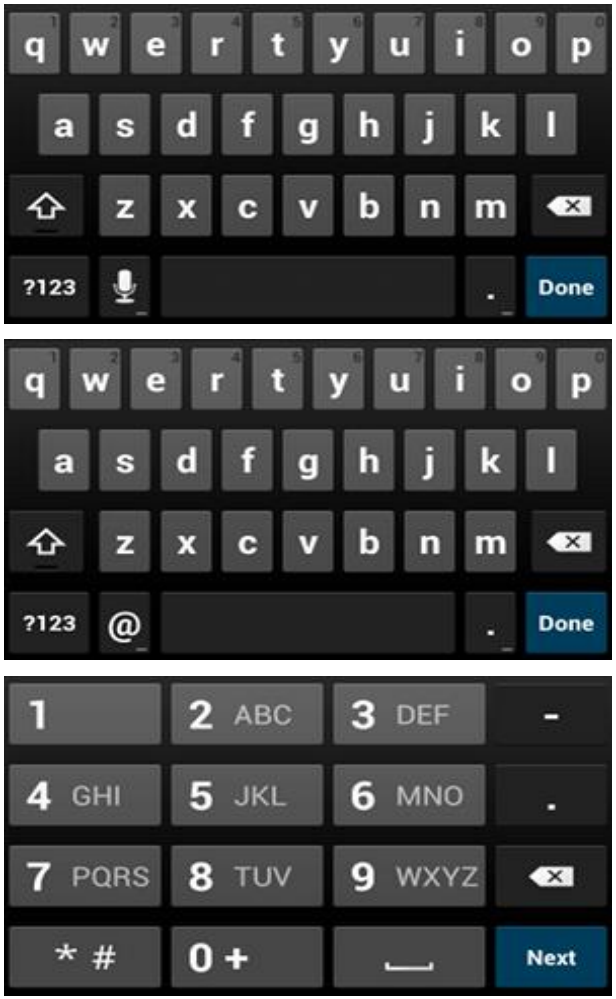
"text"
Normal text keyboard.

"textEmailAddress"
Normal text keyboard with the @ character.

"textUri"
Normal text keyboard with the / character.

"number"
Basic number keypad.

"phone"
Phone-style keypad.



Text Fields

`"textCapSentences"`
Normal text keyboard that capitalizes the first letter for each new sentence.

`"textCapWords"`
Normal text keyboard that capitalizes every word. Good for titles or person names.

`"textAutoCorrect"`
Normal text keyboard that corrects commonly misspelled words.

`"textPassword"`
Normal text keyboard, but the characters entered turn into dots.

`"textMultiLine"`
Normal text keyboard that allow users to input long strings of text that include line breaks (carriage returns).

Text Fields

- Combining multiple input types

```
<EditText
    android:id="@+id/postal_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/postal_address_hint"
    android:inputType="textPostalAddress|
                       textCapWords|
                       textNoSuggestions" />
```

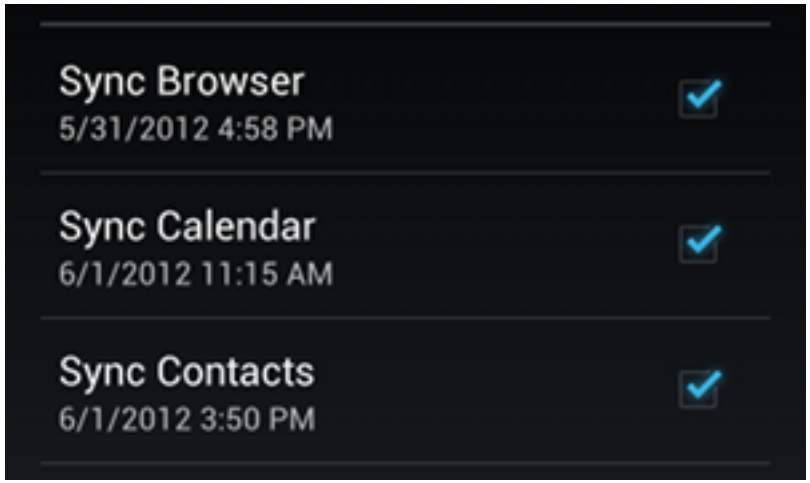
Text Fields

```
<EditText
    android:id="@+id/search"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/search_hint"
    android:inputType="text"
    android:imeOptions="actionSend" />
```


```
EditText editText = (EditText) findViewById(R.id.search);
editText.setOnEditorActionListener(new OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEND) {
            sendMessage();
            handled = true;
        }
        return handled;
    }
});
```

Check Box

- Checkboxes allow the user to select one or more options from a set.



- Present each checkbox option in a vertical list.

https://developer.android.com/guide/topics/ui/controls/checkbox

Check Box

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>

    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>

</LinearLayout>
```

Check Box

Kotlin

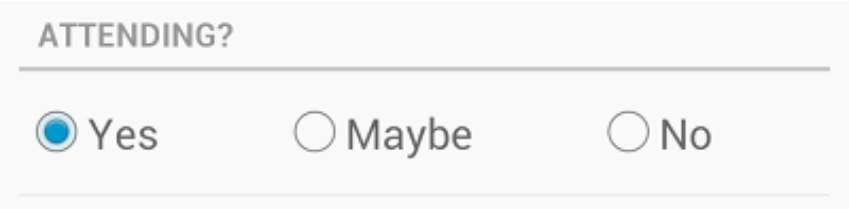
```kotlin
fun onCheckboxClicked(view: View) {
    if (view is CheckBox) {
        val checked: Boolean = view.isChecked
        when (view.id) {
            R.id.checkbox_meat -> {
                if (checked) {
                    // Put some meat on the sandwich
                } else {
                    // Remove the meat
                }
            }
            ...
```

Java

```java
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();
    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
                // Remove the meat
            break;
        . . .
```

Radio Button

- Allows user to select one option from a set.



- Use it for optional sets that are mutually exclusive if you think that the user needs to see all available options side-by-side.

https://developer.android.com/guide/topics/ui/controls/radiobutton

Radio Button

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pirates"
        android:onClick="onRadioButtonClicked"/>

    <RadioButton android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ninjas"
        android:onClick="onRadioButtonClicked"/>

</RadioGroup>
```

Radio Button

Kotlin

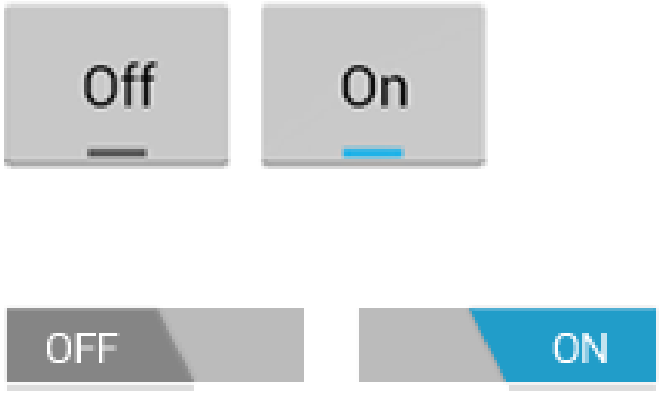
```kotlin
fun onRadioButtonClicked(view: View) {
    if (view is RadioButton) {
        // Is the button now checked?
        val checked = view.isChecked
        // Check which radio button was clicked
        when (view.getId()) {
            R.id.radio_pirates ->
                if (checked) {
                    // Pirates are the best
                }
            . . .
        }
```

Java

```java
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();
    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radio_pirates:
            if (checked)
                // Pirates are the best
            break;
        . . .
    }
}
```

Toggle Buttons

- A toggle button allows the user to change a setting between two states.



- When the user selects a ToggleButton and Switch, the object receives an on-click event.

https://developer.android.com/guide/topics/ui/controls/togglebutton

Toggle Buttons

Kotlin

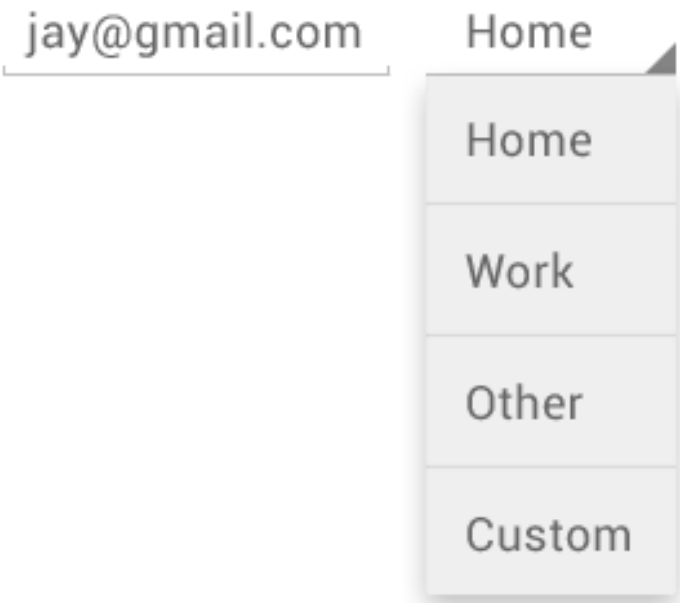
```kotlin
val toggle: ToggleButton = findViewById(R.id.togglebutton)

toggle.setOnCheckedChangeListener { _, isChecked ->
    if (isChecked) {
        // The toggle is enabled
    } else {
        // The toggle is disabled
    }
}
```

Java

```java
ToggleButton toggle = (ToggleButton) findViewById(R.id.togglebutton);

toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            // The toggle is enabled
        } else {
            // The toggle is disabled
        }
    }
});
```

Spinners

- Provide a quick way to select one value from a set
- By default, a spinner shows its currently selected value.
- Displays a dropdown menu with all other available values

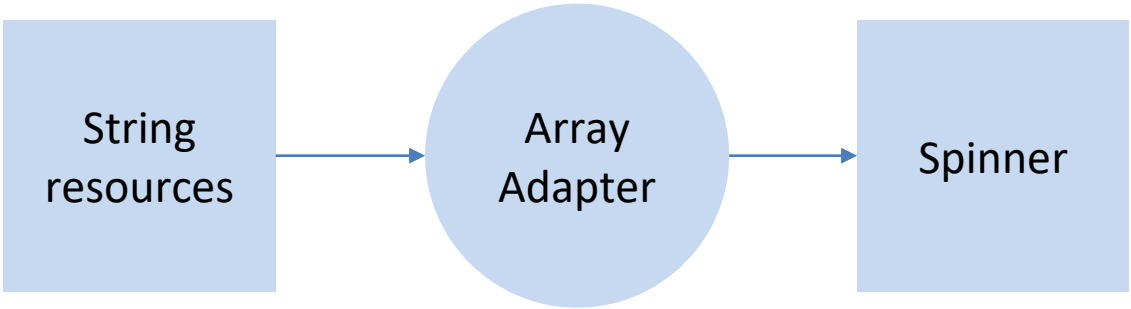


https://developer.android.com/guide/topics/ui/controls/spinner

Spinners

- Create a `string-array`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        . . .
    </string-array>
</resources>
```



Spinners – Setting adapter

Kotlin

```kotlin
val spinner: Spinner = findViewById(R.id.spinner)
// Create an ArrayAdapter using the string array and a default spinner layout
ArrayAdapter.createFromResource(
        this,
        R.array.planets_array,
        android.R.layout.simple_spinner_item
).also { adapter ->
    // Specify the layout to use when the list of choices appears
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
    // Apply the adapter to the spinner
    spinner.setAdapter = adapter
}
```

Java

```java
Spinner spinner = (Spinner) findViewById(R.id.spinner);
// Create an ArrayAdapter using the string array and a default spinner layout
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
        R.array.planets_array, android.R.layout.simple_spinner_item);
// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

Spinners – Responding to selection

Kotlin

```
class SpinnerActivity : Activity(), AdapterView.OnItemSelectedListener {
    ...
    val spinner: Spinner = findViewById(R.id.spinner)
    spinner.onItemSelectedListener = this
    ...

    override fun onItemSelected(parent: AdapterView<*>, view: View, pos: Int, id: Long) {
        // An item was selected. You can retrieve the selected item using
        // parent.getItemAtPosition(pos)
    }

    override fun onNothingSelected(parent: AdapterView<*>) {
        // Another interface callback
    }
}
```

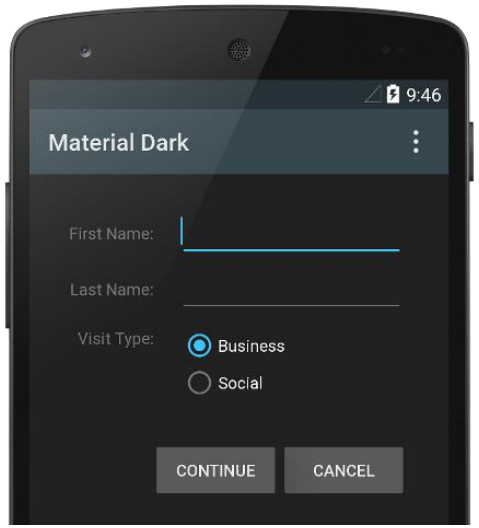
Spinners – Responding to selection

Java

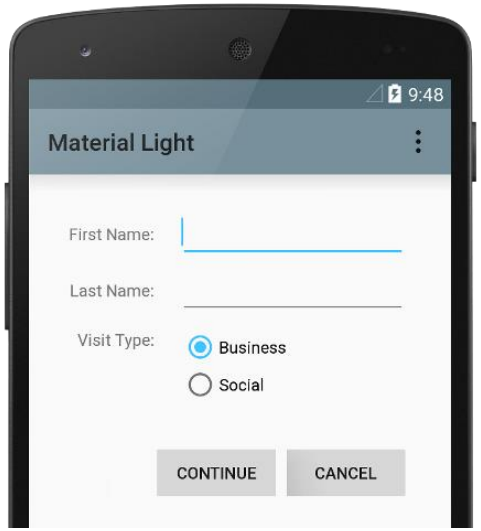
```
public class SpinnerActivity extends Activity implements OnItemSelectedListener {

    ...
    Spinner spinner = (Spinner) findViewById(R.id.spinner);
    spinner.setOnItemSelectedListener(this);
    ...

    public void onItemSelected(AdapterView<?> parent, View view,
            int pos, long id) {
        // An item was selected. You can retrieve the selected item using
        // parent.getItemAtPosition(pos)
    }

    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
}
```

Style and Theme

- Use the Android's style and theme resources to change appearance of action bar.



Dark material theme



Light material theme

https://developer.android.com/guide/topics/ui/look-and-feel/themes

Use an Android Theme

- You can apply these themes to your entire app or to individual activities by declaring them in your **manifest file** with the android:theme attribute for the <application> element or individual <activity> elements.

```
<application android:theme="@android:style/Theme.Holo.Light" ... />
```

Customize the Background

- Create a custom theme for your activity that overrides the actionBarStyle property.

- This property points to another style in which you can override the background property to specify a drawable resource for the action bar background.

Customize the Background

1. File Location: res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
           parent="@style/Theme.Holo.Light.DarkActionBar">
        <item name="android:actionBarStyle">@style/MyActionBar</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
           parent="@style/Widget.Holo.Light.ActionBar.Solid.Inverse">
        <item name="android:background">@drawable/actionbar_background</item>
    </style>
</resources>
```

2. Then apply your theme to your entire app or individual activities:

```
<application android:theme="@style/CustomActionBarTheme" ... />
```

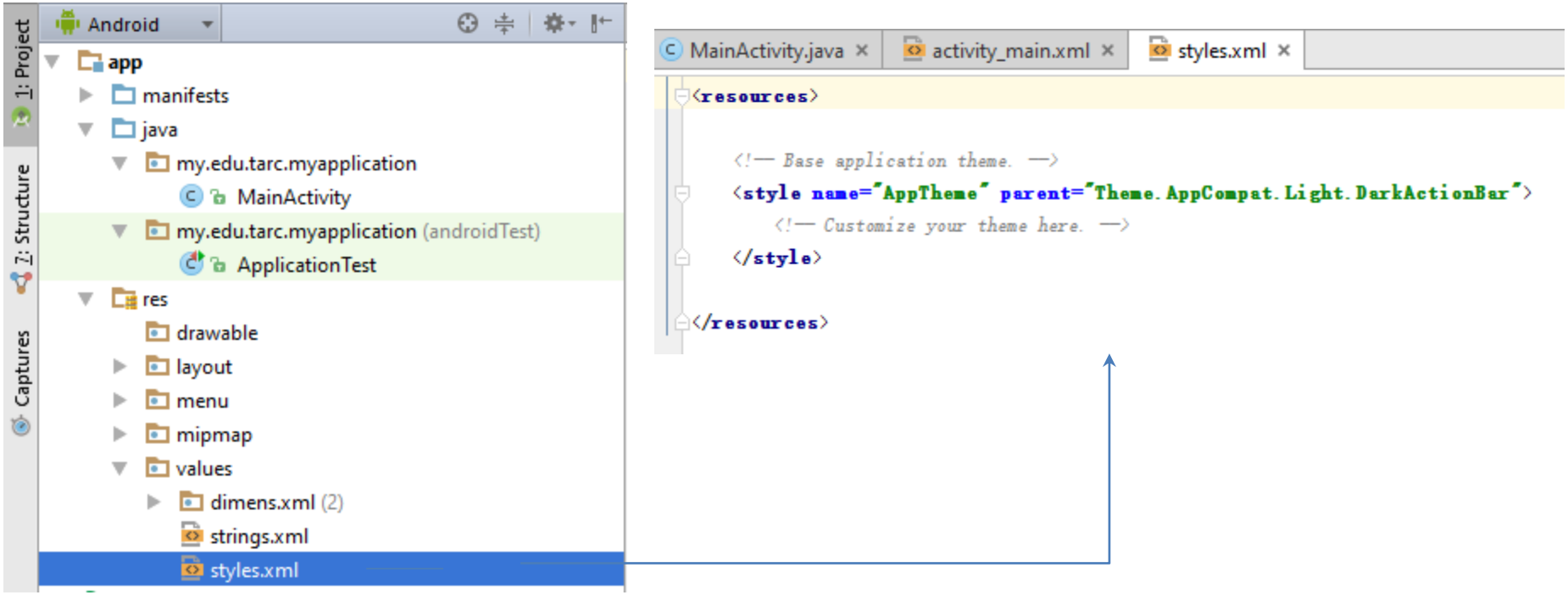
Customize the Text Color

- Action bar title: Create a custom style that specifies the textColor property and specify that style for the titleTextStyle property in your custom actionBarStyle.

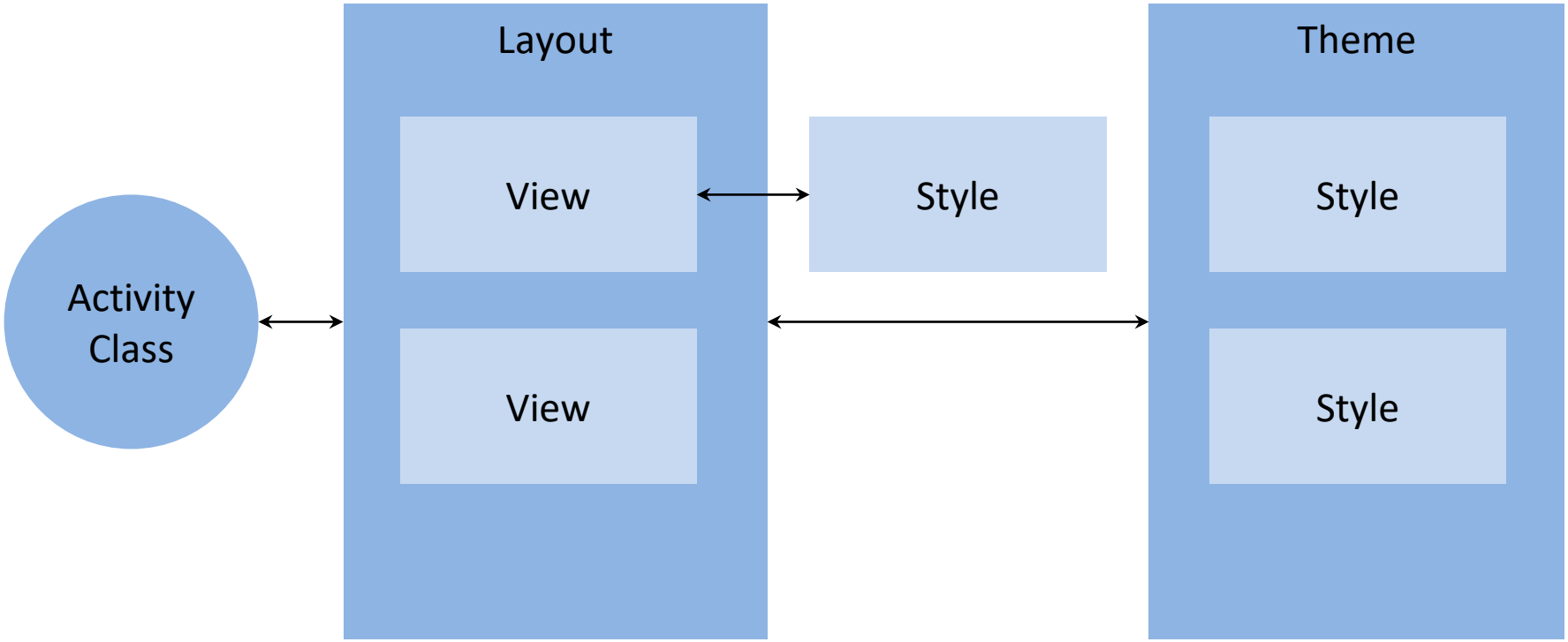
- Action bar tabs:
  Override actionBarTabTextStyle in your activity theme.

- Action buttons: Override actionMenuTextColor in your activity theme.

Styles


1: Project
7: Structure
Captures
Android
app
manifests
java
my.edu.tarc.myapplication
MainActivity
my.edu.tarc.myapplication (androidTest)
ApplicationTest
res
drawable
layout
menu
mipmap
values
dimens.xml (2)
strings.xml
styles.xml
MainActivity.java
activity_main.xml
styles.xml
<resources>
<!— Base application theme. —>
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
<!— Customize your theme here. —>
</style>
</resources>

Style vs. Theme



Dark Theme

- Available in Android 10 (API level 29) and higher

- Benefits
  - Reduces power usage
  - Improves visibility for users
  - Easier to use in a low-light environment

https://developer.android.com/guide/topics/ui/look-and-feel/darktheme

Dark Theme

- Support Dark theme in your app by:
  - Set your app's theme to inherit from a `DayNight` theme

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
```

Or

```
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
```

Review Questions

1. What are the two basic objects of an Android UI?
2. What is the main difference between a Relative and a Linear Layout?
3. What is the purpose of an adapter in a layout with ListView or GridView?
4. Differentiate Style and Theme.