Handheld Application Development

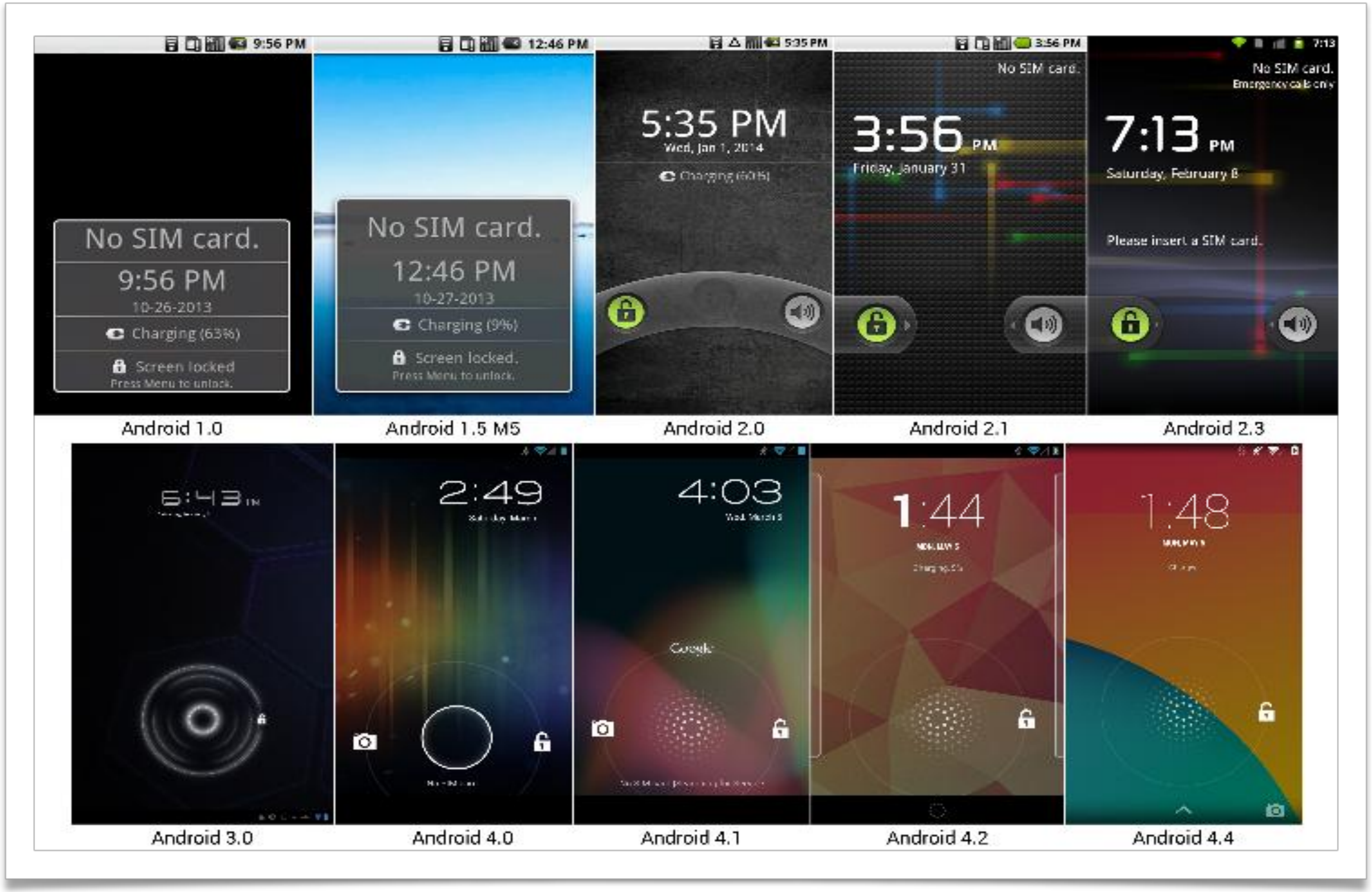
Lec 2: Layout & GUI

Ekarat Rattagan, PhD

Outline (1/2)

- Layouts
- Input Controls
- Input Events

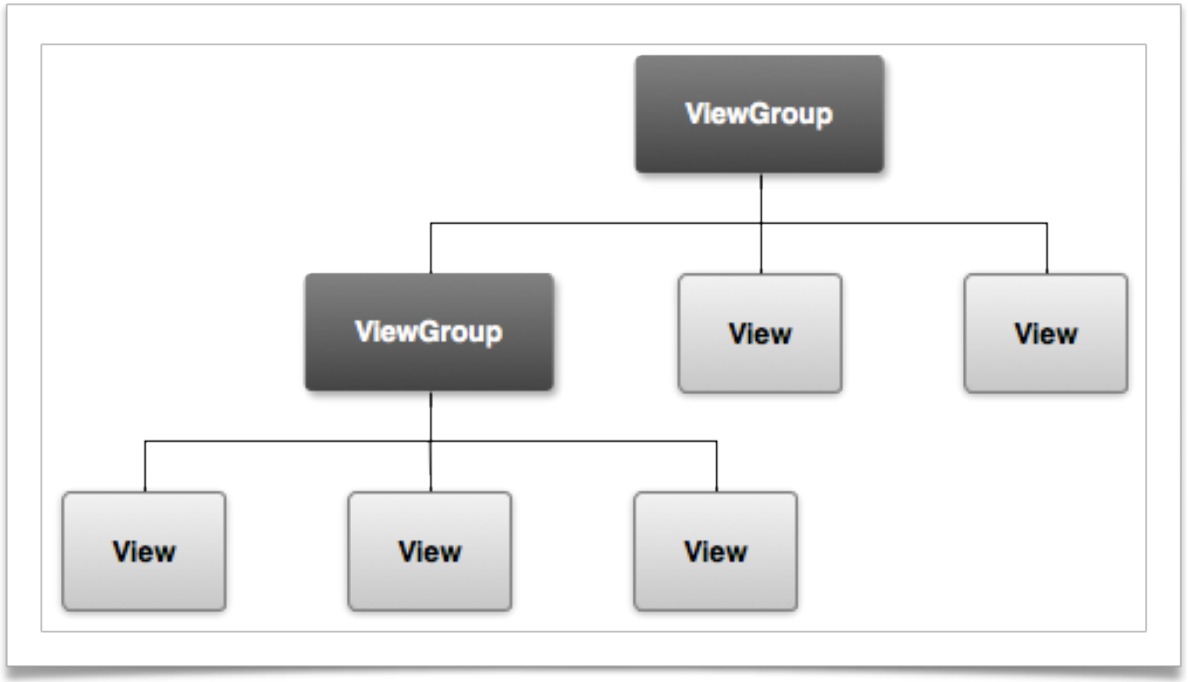
Outline (2/2)



Android 1.0
Android 1.5 M5
Android 2.0
Android 2.1
Android 2.3
Android 3.0
Android 4.0
Android 4.1
Android 4.2
Android 4.4

User Interface (UI)

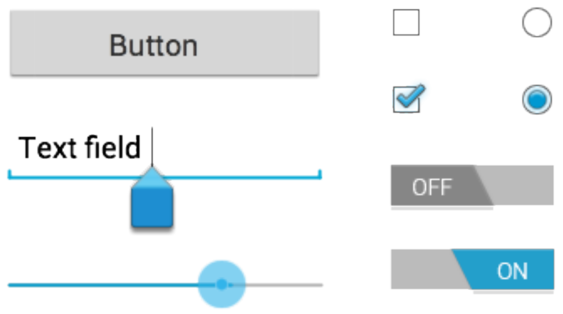
- Everything that users can see and interact with
- All UI elements are built using View and ViewGroup



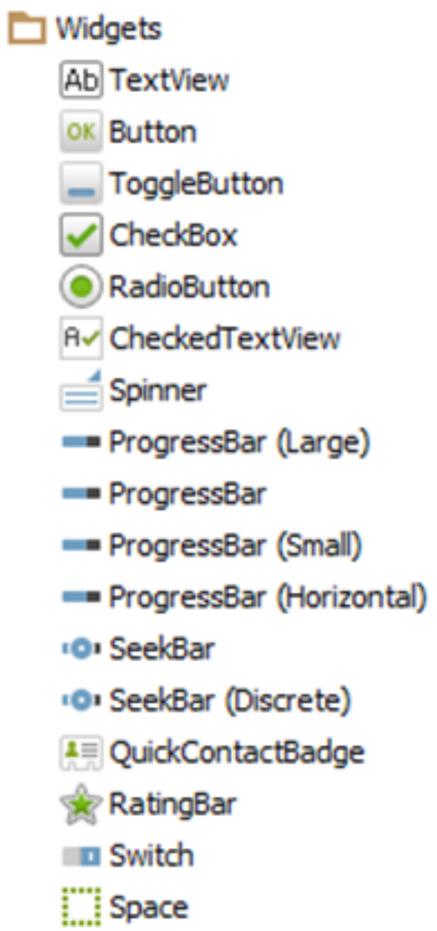
User Interface (UI)

1. Layout models, e.g., absolute, linear, relative

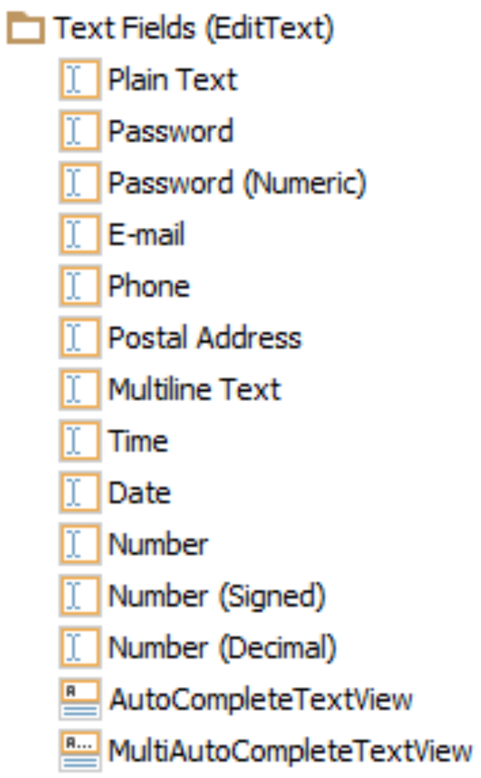
2. Input controls, e.g., buttons and text fields



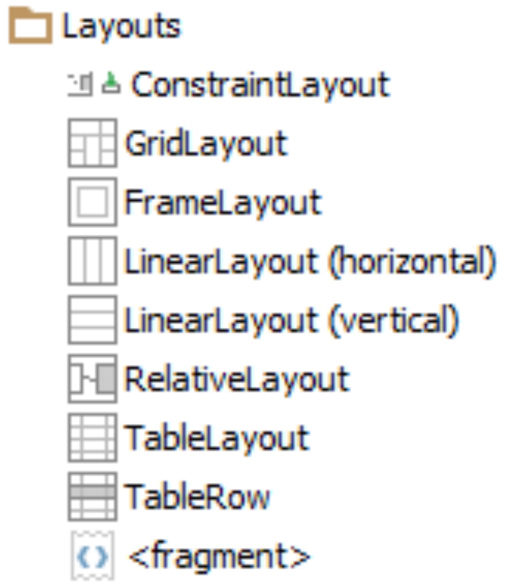
More in Android studio



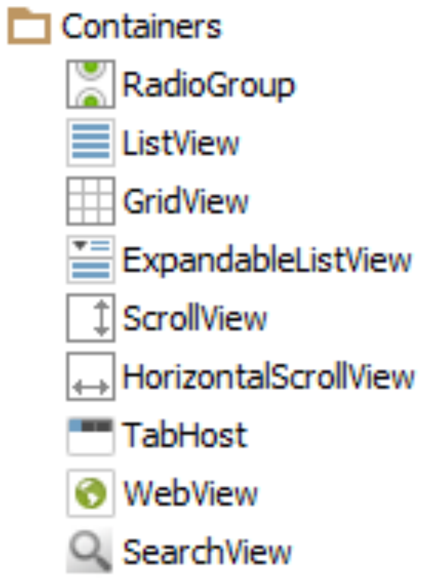
- Widgets
  - TextView
  - Button
  - ToggleButton
  - CheckBox
  - RadioButton
  - CheckedTextView
  - Spinner
  - ProgressBar (Large)
  - ProgressBar
  - ProgressBar (Small)
  - ProgressBar (Horizontal)
  - SeekBar
  - SeekBar (Discrete)
  - QuickContactBadge
  - RatingBar
  - Switch
  - Space



- Text Fields (EditText)
  - Plain Text
  - Password
  - Password (Numeric)
  - E-mail
  - Phone
  - Postal Address
  - Multiline Text
  - Time
  - Date
  - Number
  - Number (Signed)
  - Number (Decimal)
  - AutoCompleteTextView
  - MultiAutoCompleteTextView



- Layouts
  - ConstraintLayout
  - GridLayout
  - FrameLayout
  - LinearLayout (horizontal)
  - LinearLayout (vertical)
  - RelativeLayout
  - TableLayout
  - TableRow
  - <fragment>



- Containers
  - RadioGroup
  - ListView
  - GridView
  - ExpandableListView
  - ScrollView
  - HorizontalScrollView
  - TabHost
  - WebView
  - SearchView

1. Layout models

Two ways of creation

1. Declare UI elements in XML
- Using Android's XML vocabulary

2. Instantiate layout elements at runtime
- Create View and ViewGroup objects programmatically.

1. Layout models

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

}
```

1. **Declare UI elements in XML**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
  <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
  <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

1. Layout models

```
//Creating LinearLayout.
 LinearLayout linearlayout = new LinearLayout(this);

 //Setting up LinearLayout Orientation
 linearlayout.setOrientation(LinearLayout.VERTICAL);

 LayoutParams linearlayoutlayoutparams = new
LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT);

 setContentView(linearlayout, linearlayoutlayoutparams)
```

1. Layout models

Type of layouts

- 1.1 Absolute
- 1.2 Linear
- 1.3 Relative

1.1 Absolute Layout

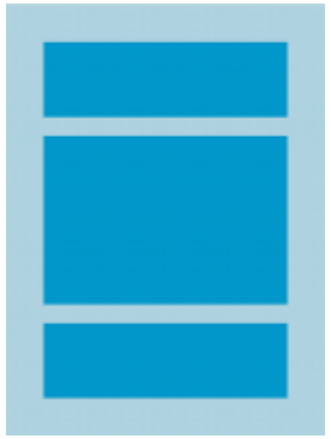
Enables you to specify the exact location of its children

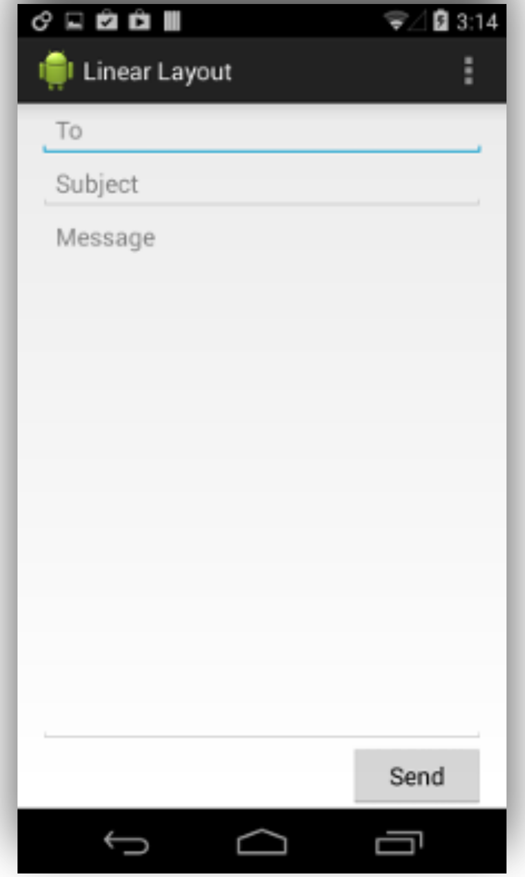
```
<AbsoluteLayout
        android:layout_width=”fill_parent”
        android:layout_height=”fill_parent”
        xmlns:android=”http://schemas.android.com/apk/res/android” >
        <Button
                android:layout_width=”188dp”
                android:layout_height=”wrap_content”
                android:text=”Button”
                android:layout_x=”126px”
                android:layout_y=”361px” />
</AbsoluteLayout>
```

More about dp, https://www.captechconsulting.com/blogs/understanding-density-independence-in-android

1.2 Linear Layout

A view group that aligns all children in a single direction, vertically or horizontally.





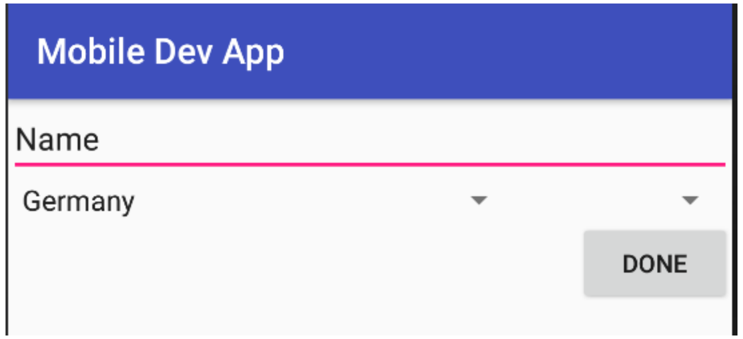
```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:orientation="vertical" >
```

1.3 Relative Layout (1/3)

A view group that displays child views in relative positions.

- Relative to sibling elements, e.g., left-of or below another view
- Relative to the parent

1.3 Relative Layout (2/3)



android:layout_alignParentTop
If "true", makes the top edge of this view match the top edge of the parent.

android:layout_centerVertical
If "true", centers this child vertically within its parent.

android:layout_below
Positions the top edge of this view below the view specified with a resource ID.

android:layout_toRightOf
Positions the left edge of this view to the right of the view specified with a resource ID.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/
apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Week2_UI">

    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        tools:layout_editor_absoluteX="64dp"
        tools:layout_editor_absoluteY="14dp" />

    <Spinner
        android:id="@+id/dates"
        android:layout_width="260dp"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/name"
        android:layout_toLeftOf="@+id/times"
        android:entries="@array/team"
        />

    <Spinner
        android:id="@+id/times"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/name" />

    <Button
        android:id="@+id/button6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/times"
        android:text="@string/done" />
</RelativeLayout>
```

1.3 Relative Layout (3/3)

A very powerful utility for designing a user interface

- Can eliminate nested view groups and keep your layout hierarchy flat
  - Improves performance.
  - If several nested LinearLayout groups, replace with a single RelativeLayout.

2. Input controls

Control Type	Description	Related Classes
2.1 Button	A push-button that can be pressed, or clicked, by the user to perform an action.	Button
2.2 Text field	An editable text field. You can use the AutoCompleteTextView widget to create a text entry widget that provides auto-complete suggestions	EditText, AutoCompleteTextView
2.3 Checkbox	An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.	CheckBox
2.4 Radio button	Similar to checkboxes, except that only one option can be selected in the group.	RadioGroup RadioButton
2.5 Toggle button	An on/off button with a light indicator.	ToggleButton
2.6 Spinner	A drop-down list that allows users to select one value from a set.	Spinner
2.7 Pickers	A dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture. Use a DatePickercode> widget to enter the values for the date (month, day, year) or a TimePicker widget to enter the values for a time (hour, minute, AM/PM), which will be formatted automatically for the user's locale.	DatePicker,TimePicker

2.1 Button

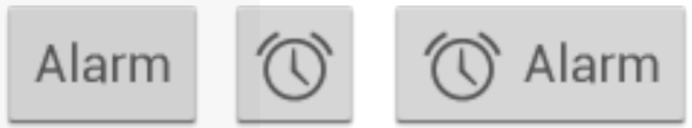
A button consists of text and/or an icon that communicates what action occurs when the user touches it.

With text, using the `Button` class:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

With an icon, using the `ImageButton` class:

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```



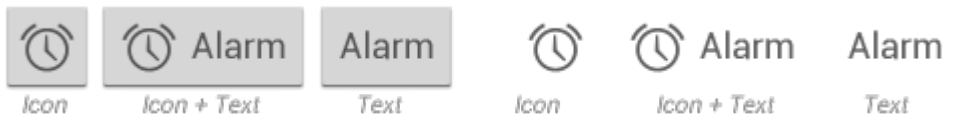
With text and an icon, using the `Button` class with the `android:drawableLeft` attribute:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:drawableLeft="@drawable/button_icon"
    ... />
```

2.1 More Button Styling

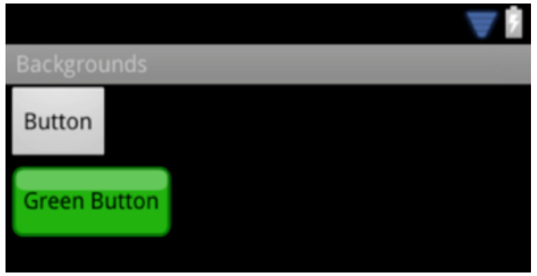
1. Borderless button

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage"
    style="?android:attr/borderlessButtonStyle" />
```



2. Custom background

[Right click] Drawable > New > Drawable resource file



```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/
res/android">
    <item android:drawable="@color/colorAccent"
    android:state_pressed="true" />
    <item android:drawable="@color/colorPrimary"
    android:state_focused="true" />
    <item android:drawable="@color/colorPrimaryDark" />
</selector>
```

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage"
    android:background="@drawable/button_custom"
 />
```

2.1 Button (click events)

- Using an OnClickListener

```
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
    }
});
```

Input Events

- Other register options
  - 2nd option

3rd option

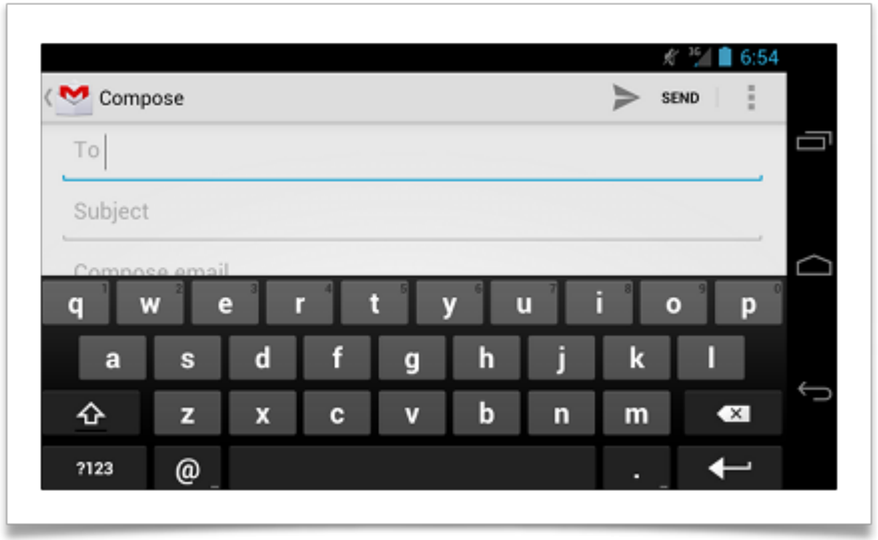
```java
private OnClickListener mListener = new
OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {

   Button button = (Button)findViewById(R.id.btn);
   button.setOnClickListener(mListener);

}
```

```java
public class ExampleActivity extends Activity
implements OnClickListener {

   protected void onCreate(Bundle savedValues) {
        ...
     Button button = findViewById(R.id.corky);
     button.setOnClickListener(this);
   }

   public void onClick(View v) {
      // do something when the button is clicked
   }
   ...
}
```

2.2 Text Fields

Allow users to type text into an app

- Single and Multiple line
- Touching a text field
  - Place the cursor
  - Automatically displays the keyboard



2.2 Text Fields

Keyboard Type

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

`"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

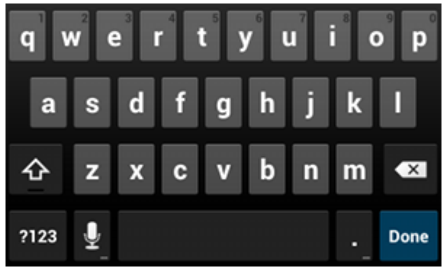


Figure 1. The default `text` input type.

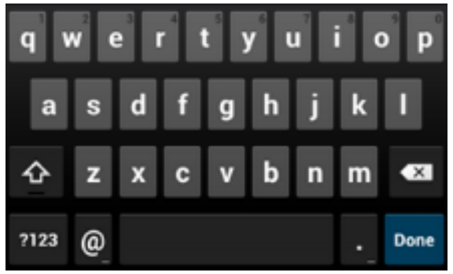


Figure 2. The `textEmailAddress` input type.

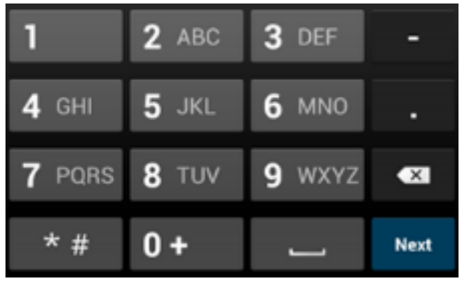
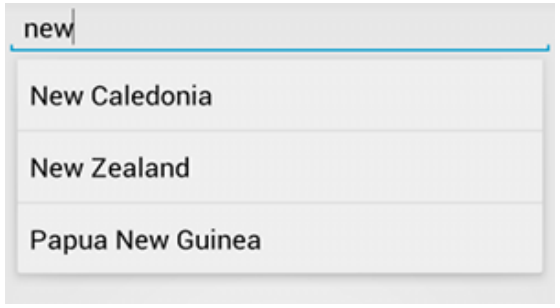


Figure 3. The `phone` input type.

2.2 Text Fields

Auto-complete suggestions



```xml
<?xml version="1.0" encoding="utf-8"?>
<AutoCompleteTextView xmlns:android=
"http://schemas.android.com/apk/res/
android"
    android:id="@+id/autocomplete_country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

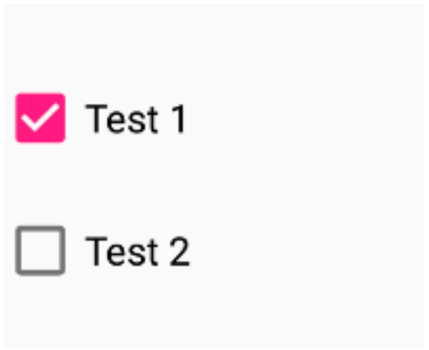
```java
// Get a reference to the AutoCompleteTextView in the layout
AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.autocomplete_country);
// Get the string array
String[] countries =
getResources().getStringArray(R.array.countries_array);
// Create the adapter and set it to the AutoCompleteTextView
ArrayAdapter<String> adapter =
        new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, countries);
textView.setAdapter(adapter);
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries_array">
        <item>Afghanistan</item>
        <item>Albania</item>
        <item>Algeria</item>
        <item>American Samoa</item>
        <item>Andorra</item>
        <item>Angola</item>
        <item>Anguilla</item>
        <item>Antarctica</item>
        ...
    </string-array>
</resources>
```

2.3 Checkboxes

Allow the user to select one or more options from a set

- Present in a vertical list



```
<CheckBox
    android:id="@+id/checkBox2"
    android:layout_width="86dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="182dp"
    android:text="Test 2" />

<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="88dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="230dp"
    android:text="Test 1" />
```

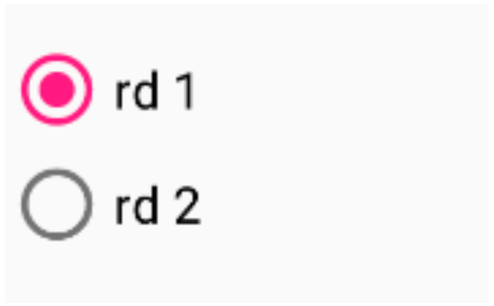
2.3 Checkboxes (Code examples)

```
CheckBox ch1, ch2;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_week2__ui);
    ch1 = findViewById(R.id.checkBox1);
    ch2 = findViewById(R.id.checkBox2);

    ch1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Is the view now checked?

            boolean checked = ((CheckBox) view).isChecked();
            Toast.makeText(getApplicationContext(),view.getId()+" is "+checked,Toast.LENGTH_SHORT).show();
        }
    });

    ch2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Is the view now checked?

            boolean checked = ((CheckBox) view).isChecked();
            Toast.makeText(getApplicationContext(),view.getId()+" is "+checked,Toast.LENGTH_SHORT).show();
        }
    });
}
```

2.4 Radio Buttons

Allow the user to select one option from a set.

- Mutually exclusive



```
<RadioGroup
    android:id="@+id/rdg"
    android:layout_width="227dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="85dp" >

    <RadioButton
        android:id="@+id/radioButton1"
        android:layout_width="68dp"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:checked="true"
        android:text="rd 1" />

    <RadioButton
        android:id="@+id/radioButton2"
        android:layout_width="71dp"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/checkBox2"
        android:text="rd 2" />

</RadioGroup>
```

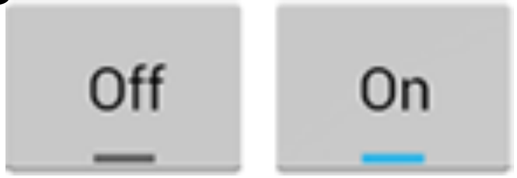
2.4 Radio Buttons (code examples)

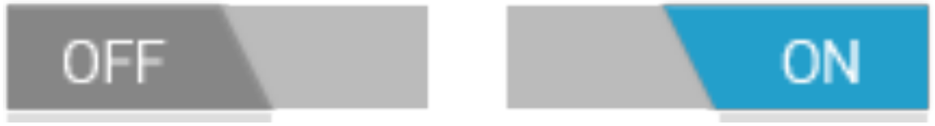
```
rdb1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        testRadioButtonCheck(view);
    }
});

rdb2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        testRadioButtonCheck(view);
    }
});
```

```
void testRadioButtonCheck(View view){
    // Is the button now checked?
    boolean checked = ((RadioButton)
view).isChecked();
    String result = "";
    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radioButton1:
            if (checked)
                result = "rd 1";
                break;
        case R.id.radioButton2:
            if (checked)
                result = "rd 2";
                break;
    }


Toast.makeText(getApplicationContext(),result,Toast
.LENGTH_SHORT).show();
}
```

2.5 Toggle Buttons

Allows the user to change a setting between two states





Android 4.0+

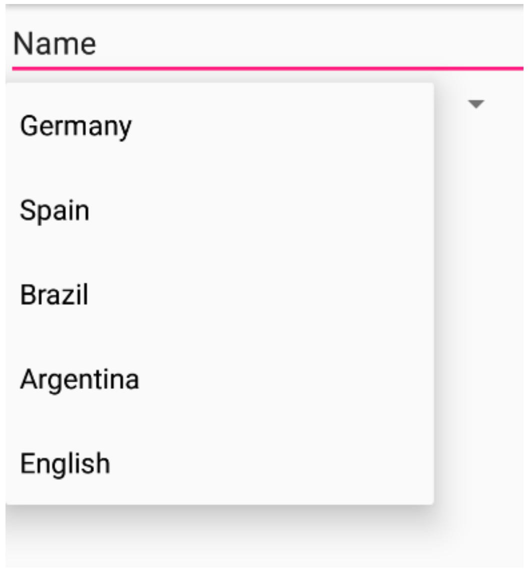
```
ToggleButton toggle = (ToggleButton) findViewById(R.id.togglebutton);
toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            // The toggle is enabled
  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

2.6 Spinners

Provide a quick way to select one value from a set

- Displays a dropdown menu

```
<Spinner
    android:id="@+id/team"
    android:layout_width="260dp"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/name"
    android:layout_toLeftOf="@+id/times"
    android:entries="@array/team"
    />
```



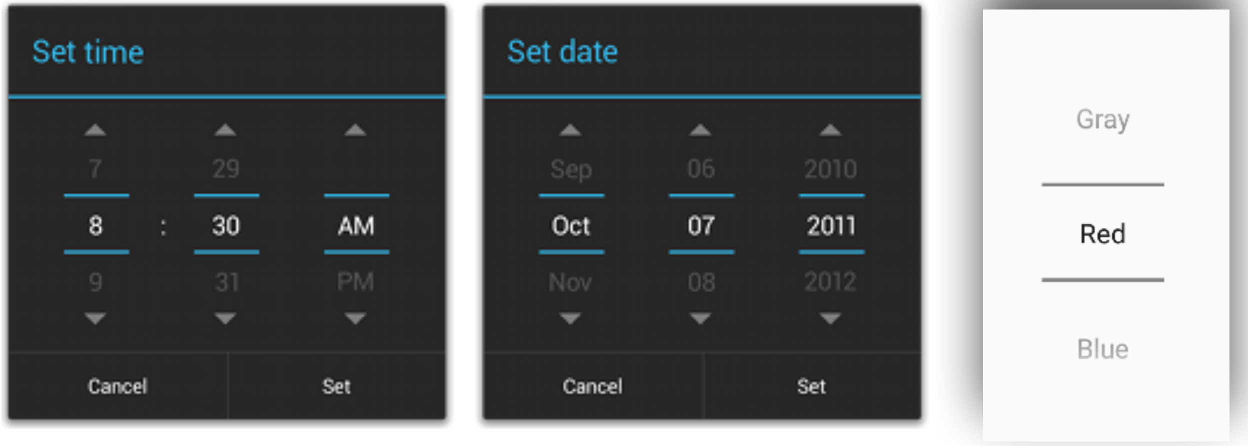
2.6 Spinners (Code examples)

```
teamSpinner = findViewById(R.id.team);

final String[] teamList = getResources().getStringArray(R.array.team);

ArrayAdapter<String> adapterThai = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, teamList);
        teamSpinner.setAdapter(adapterThai);

teamSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(Week2_UI.this,
                "Select : " + teamList[position],
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

2.7 Pickers (Android studio 3.1 bug—No UI element)

Provides controls for selecting each part of the

- Time (hour, minute, AM/PM), Date (month, day, year), Generic
- Ensure that your users can pick a time or date that is valid, formatted correctly, and adjusted to the user's locale

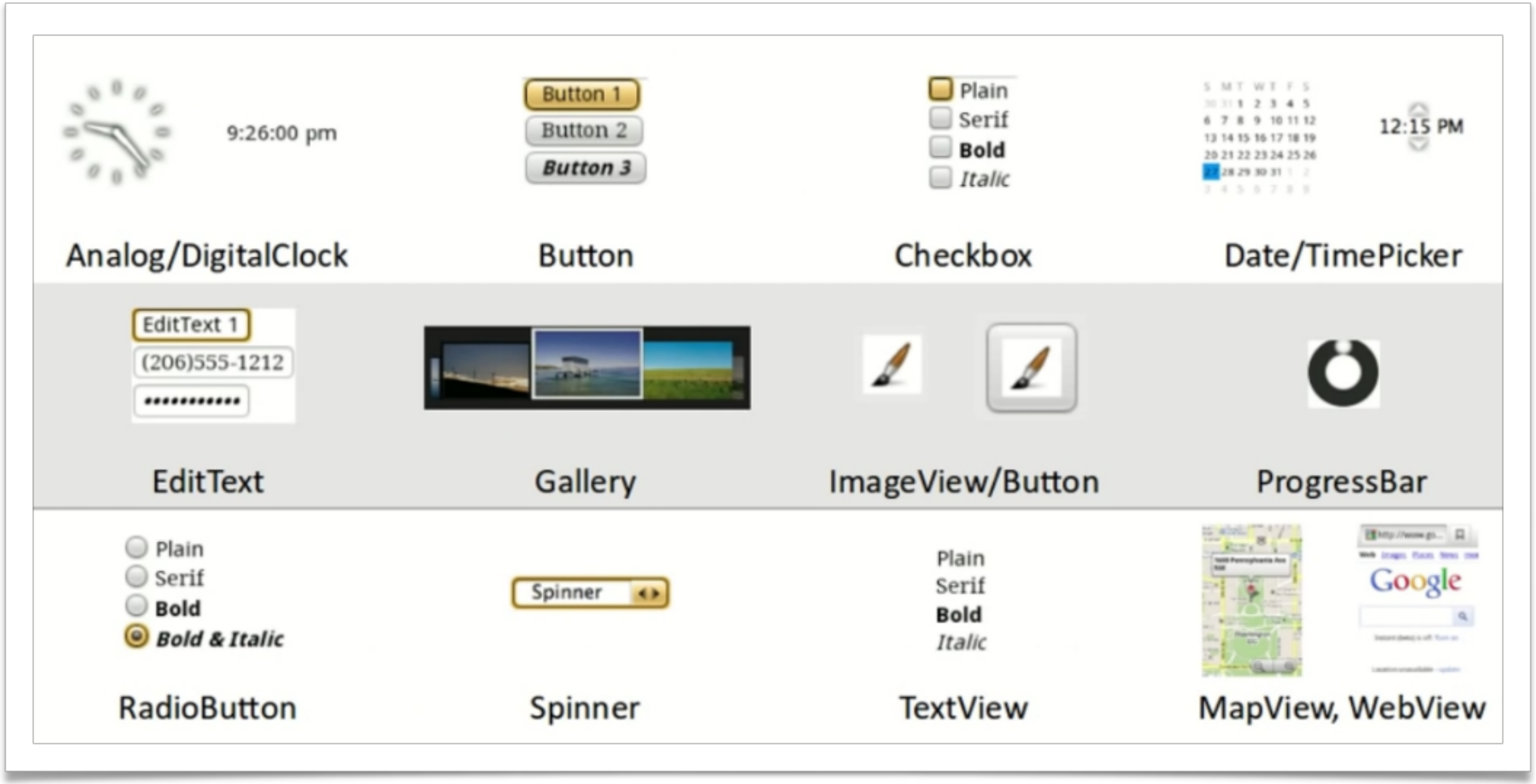


2.7 Pickers

```
NumberPicker pickers;

pickers = (NumberPicker)findViewById(R.id.numberPicker);

final String[] arrayPicker= new String[]{"Red", "Blue", "Green", "Yellow", "Gray"};

//set min value zero
pickers.setMinValue(0);

//set max value from length array string reduced 1
pickers.setMaxValue(arrayPicker.length - 1);

pickers.setOnValueChangedListener(new NumberPicker.OnValueChangeListener() {
    @Override
    public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
        //result.setText(arrayPicker[picker.getValue()]);
        String color = arrayPicker[picker.getValue()];
        Toast.makeText(getApplicationContext(),color,Toast.LENGTH_SHORT).show();

    }
});
```

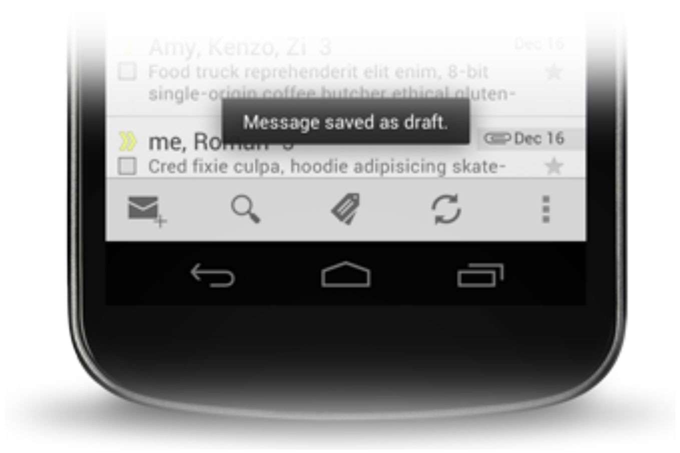
```
//Add these two lines of code
pickers.setDisplayedValues(arrayPickers);
pickers.setWrapSelectorWheel(false);
```

Android widgets



Toast

- A simple feedback about an operation in a small popup


Amy, Kenzo, Zi 3
Food truck reprehenderit elit enim, 8-bit
single-origin coffee butcher ethical gluten-
Message saved as draft.
me, Roman 3
Dec 16
Cred fixie culpa, hoodie adipisicing skate-

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Conclusion

- What you have learned
  - Layout
  - UI elements
  - Input events
  - Toasts

Resource

- http://unitid.nl/androidpatterns/uap_category/getting-input
- https://developer.android.com/guide/topics/ui/overview.html
- Library
    - https://github.com/codepath/android_guides/wiki/Must-Have-Libraries
    - https://github.com/square/leakcanary
    - https://github.com/code-troopers/android-betterpickers
    - https://github.com/wasabeef/awesome-android-ui
    - https://infinum.co/the-capsized-eight/articles/top-5-android-libraries-every-android-developer-should-know-about
    - http://blog.teamtreehouse.com/android-libraries-use-every-project
    - https://github.com/ddanny/achartengine