Mobile App Development

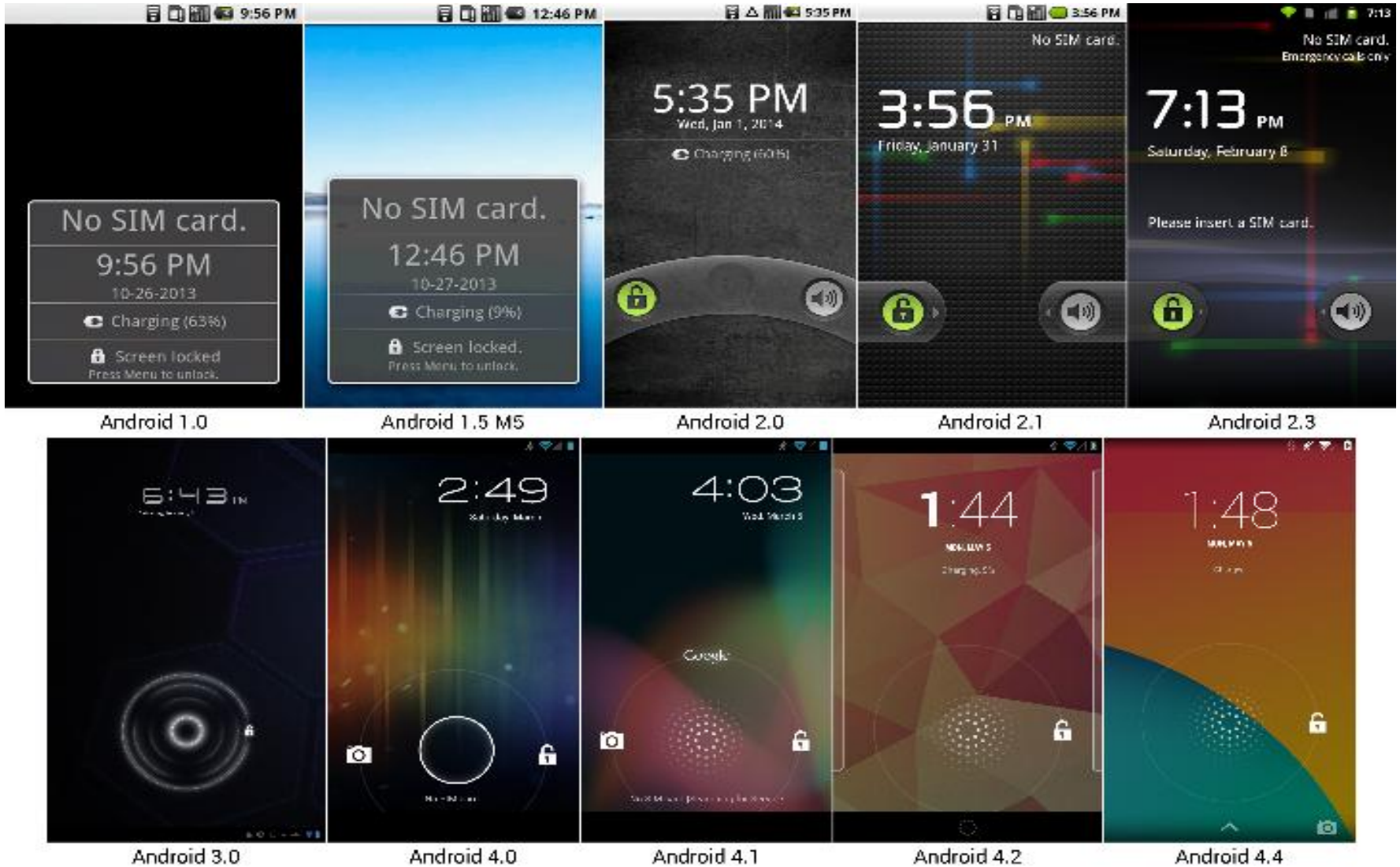
Lec2: User Interface (UI)

Ekarat Rattagan, PhD

Outline (1/2)

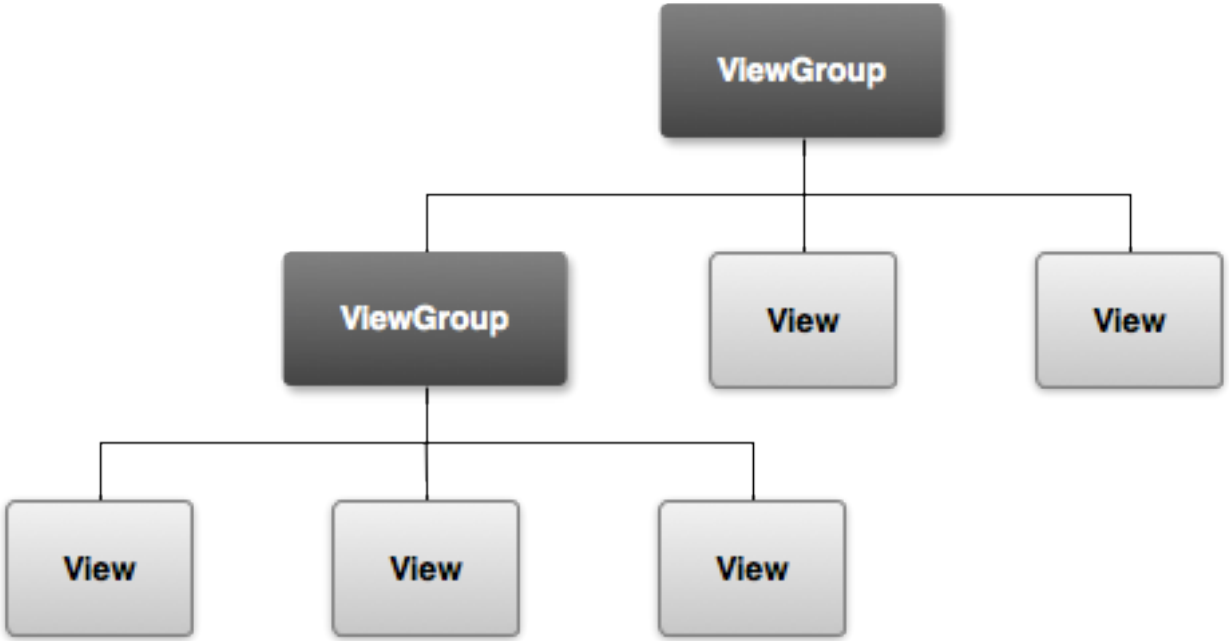
- Layouts
- Input Controls
  - Toasts
- Input Events

Outline (2/2)


9:56 PM
No SIM card.
9:56 PM
10-26-2013
Charging (63%)
Screen locked
Press Menu to unlock.
Android 1.0
12:46 PM
No SIM card.
12:46 PM
10-27-2013
Charging (9%)
Screen locked.
Press Menu to unlock.
Android 1.5 M5
5:35 PM
5:35 PM
Wed, Jan 1, 2014
Charging (60%)
Android 2.0
3:56 PM
No SIM card.
3:56 PM
Friday, January 31
Android 2.1
7:13
No SIM card.
Emergency calls only
7:13 PM
Saturday, February 8
Please insert a SIM card.
Android 2.3
6:43
Android 3.0
2:49
Android 4.0
4:03
Google
Android 4.1
1:44
Android 4.2
1:48
Android 4.4

User Interface (UI)

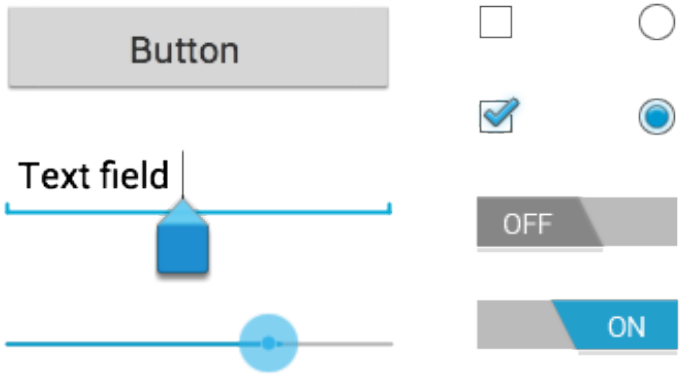
- Everything that the user can see and interact with
- All UI elements are built using
  - View and ViewGroup



User Interface (UI)

1. Layout models, e.g., a linear or relative layout

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



User Interface (UI)

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

Visual structure for a user interface

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>
```

1.3 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

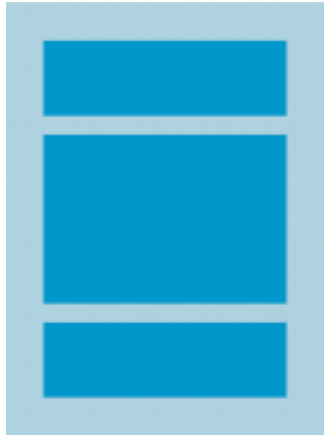
1. Layout models

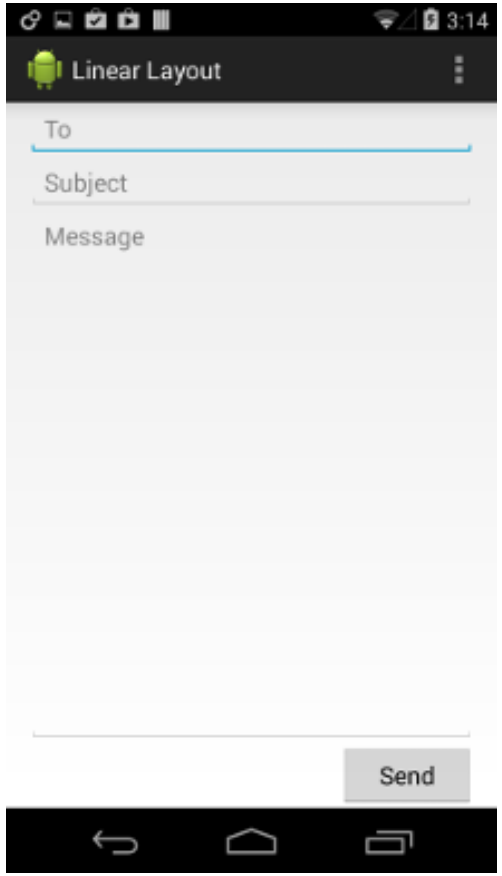
Type of layouts

1.1 Absolute
1.2 Linear
1.3 Relative

1.1 Linear Layout

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





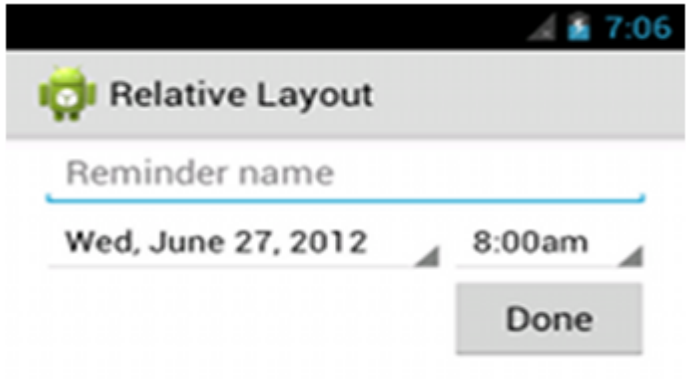

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:orientation="vertical" >
```

1.2 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

1.2 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.

```
<RelativeLayout xmlns:android="http://schemas.android.com/
apk/
res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

1.2 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:

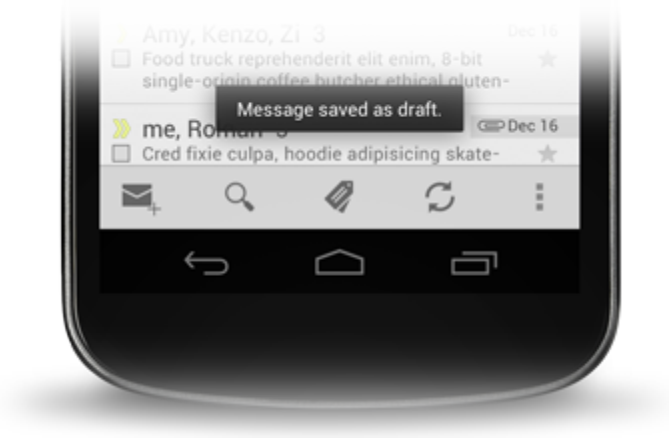
```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

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"
    ... />
```

Toast

- A simple feedback about an operation in a small popup




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

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

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

- Registered example
  - 1st option

```
private OnClickListener mCorkyListener = 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.corky);
    button.setOnClickListener(mCorkyListener);

}
```

2nd option

```
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
    }
    ...
}
```

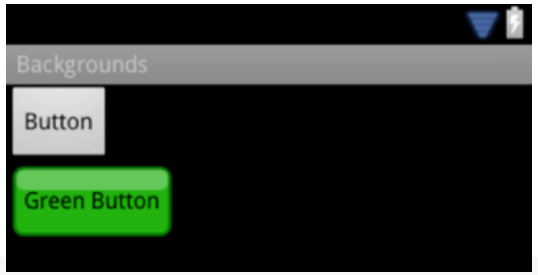
2.1 More Button Styling

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



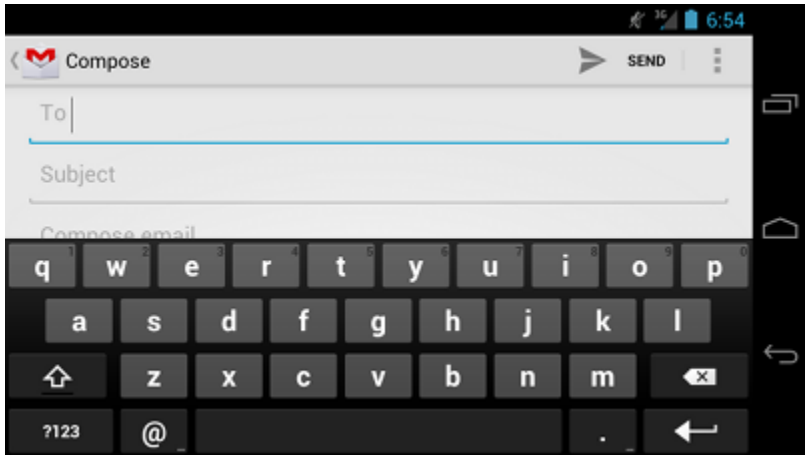

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@color/colorAccent"
android:state_pressed="true" />
    <item android:drawable="@color/colorPrimary"
android:state_focused="true" />
    <item android:drawable="@color/colorPrimaryDark"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

2.2 Text Fields

Allow the user 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

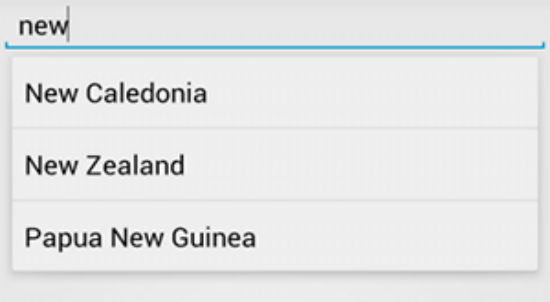
Figure 1. The default `text` input type.

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" />
```

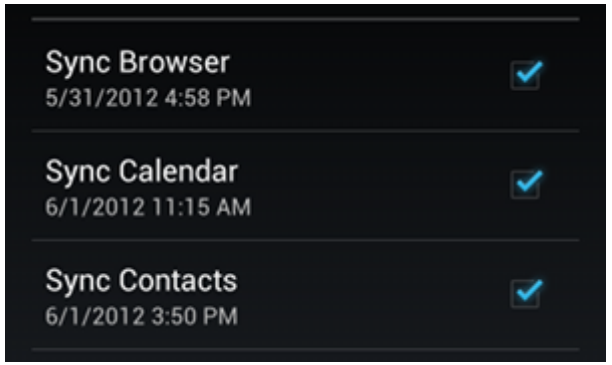
```java
// Get a reference to the AutoCompleteTextView in the
layout
AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.autocomplete_country);
// Get the string array
String[] countries =
getResources().getStringArray(R.array.countries_array);
// Create the adapter and set it to 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




```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>
```

2.3 Checkboxes

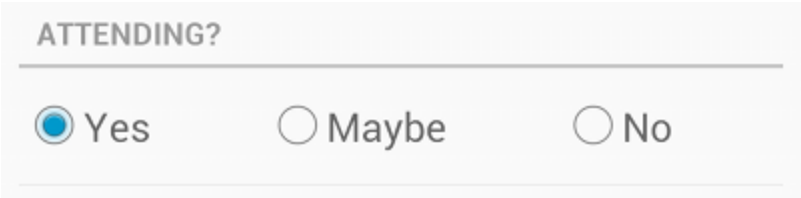
Source code example

```
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;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

2.4 Radio Buttons

Allow the user to select one option from a set.

- Mutually exclusive




```xml
<?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>
```

2.4 Radio Buttons

Source code example

```
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;
        case R.id.radio_ninjas:
            if (checked)
                // Ninjas rule
            break;
    }
}
```

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/planets_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
    </string-array>
</resources>
```

2.6 Spinners

Source code example

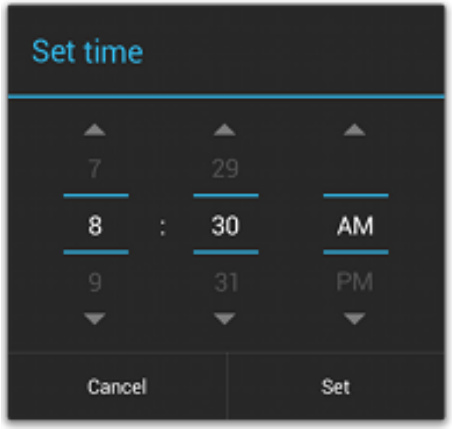
```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
                                        R.array.planets_array,
                                         android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

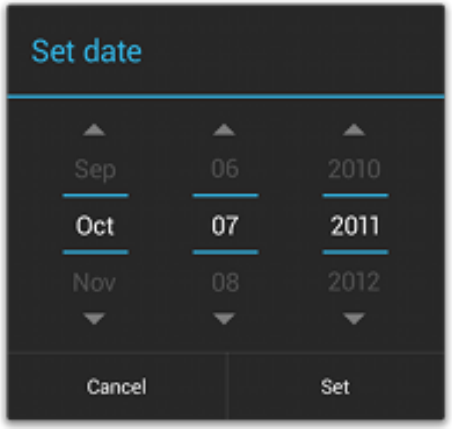
```
public class SpinnerActivity extends Activity implements OnItemSelectedListener {
    ...

    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
    }
}
```

2.7 Pickers

Provides controls for selecting each part of the

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





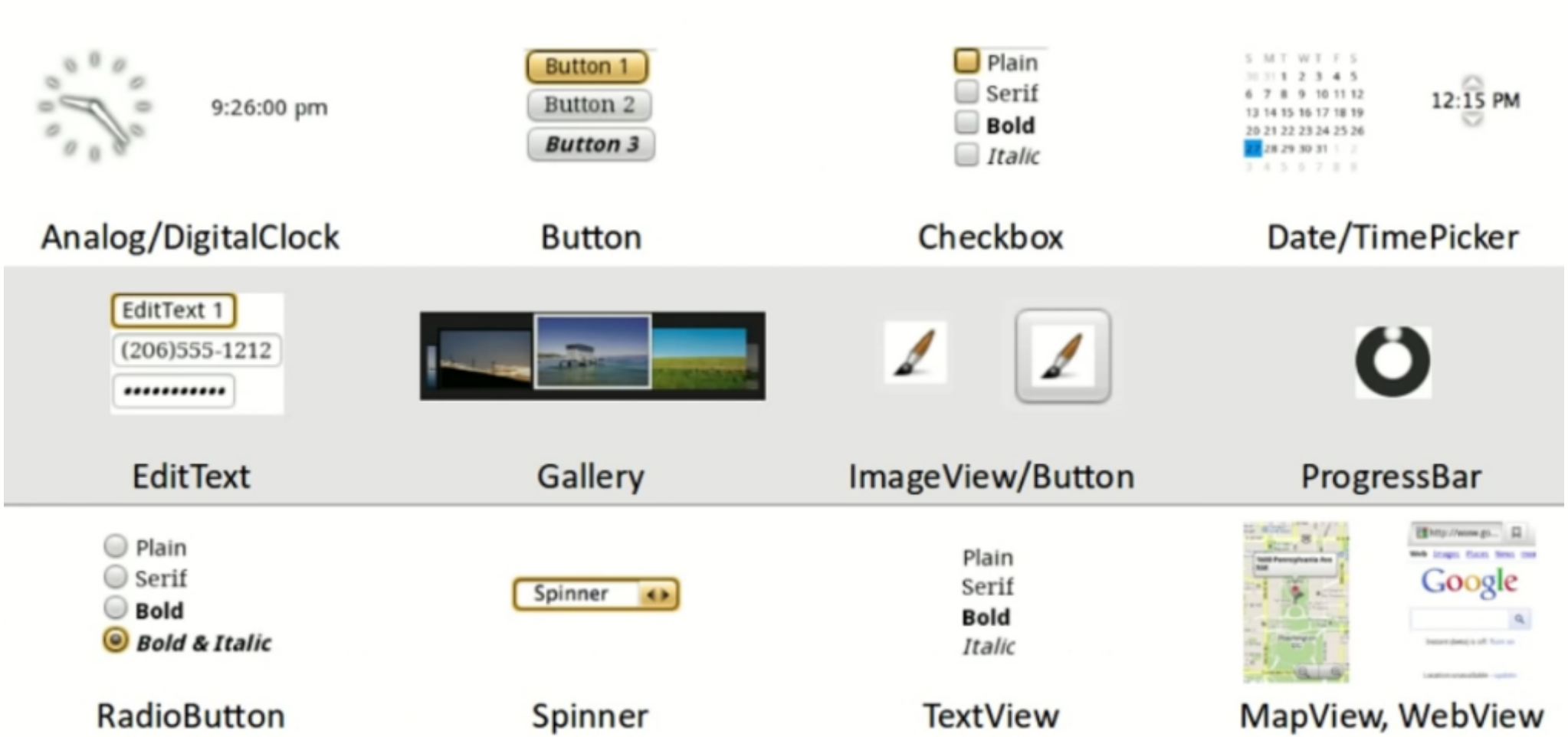
2.7 Pickers

Extending DialogFragment for a time picker

```
public static class TimePickerFragment extends DialogFragment
                            implements
TimePickerDialog.OnTimeSetListener {

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the current time as the default values for the picker
        final Calendar c = Calendar.getInstance();
        int hour = c.get(Calendar.HOUR_OF_DAY);
        int minute = c.get(Calendar.MINUTE);

        // Create a new instance of TimePickerDialog and return it
        return new TimePickerDialog(getActivity(), this, hour, minute,
                DateFormat.is24HourFormat(getActivity()));
    }

    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        // Do something with the time chosen by the user
    }
}
```

```
public void showTimePickerDialog(View v) {
    DialogFragment newFragment = new TimePickerFragment();
    newFragment.show(getSupportFragmentManager(), "timePicker");
}
```

Android widgets



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