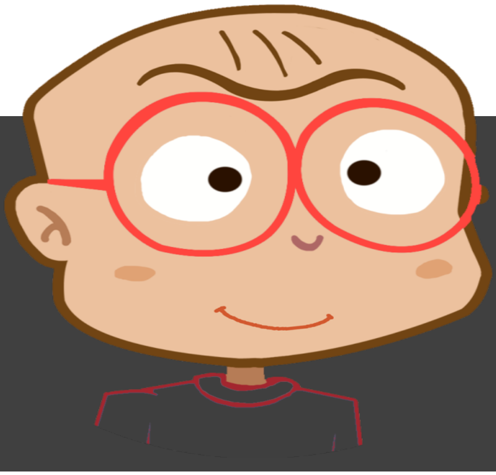
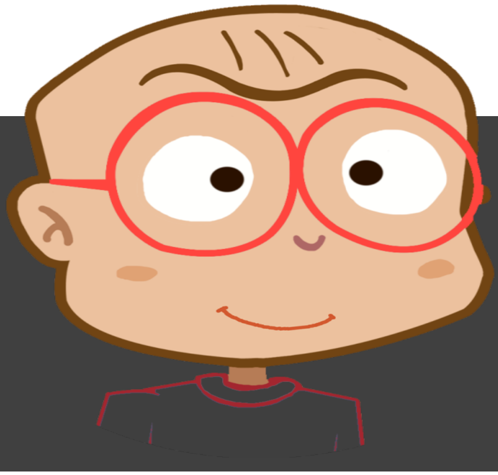


ANDROID USER INTERFACE

Basic Widget

START

Table of contents



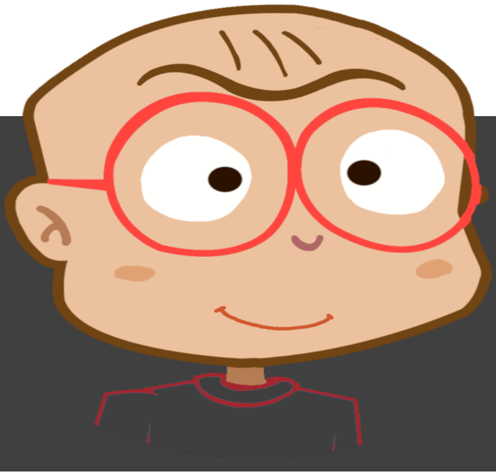
Basic widget types

Basic widget types

UI Controls	Description
TextView	This control is used to display text to the user
EditText	EditText is a predefined subclass of TextView that includes rich editing capabilities.
AutoCompleteTextView	The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.
Button/ImageButton	A push-button that can be pressed, or clicked, by the user to perform an action.
CheckBox	An on/off switch that can be toggled by the user.

Basic widget types

UI Controls	Description
ToggleButton	An on/off button with a light indicator.
RadioGroup/RadioButton	The RadioButton has two states: either checked or unchecked.
TimePicker/DatePicker	The view enables users to select a time of the day/date of the day
ProgressBar	The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.
Spinner	A drop-down list that allows users to select one value from a set.



Event Handling

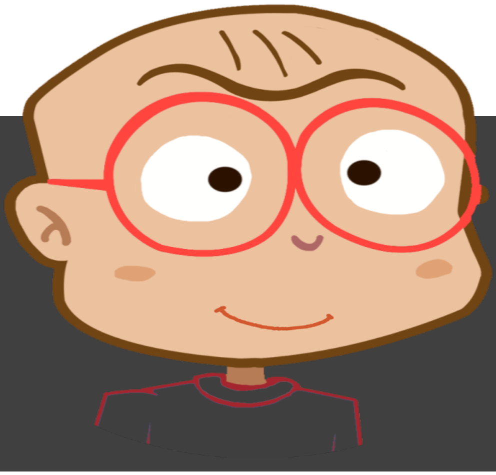
Event Handling

Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	**OnFocusChangeListener()** This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.
onKey()	**OnFocusChangeListener()** This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.

Event Handling (cont)

Event Handler	Event Listener & Description
onTouch()	**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	**OnMenuItemClickListener()** This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.

```
tvDelayTime.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {
                // Todo
        }
});
```



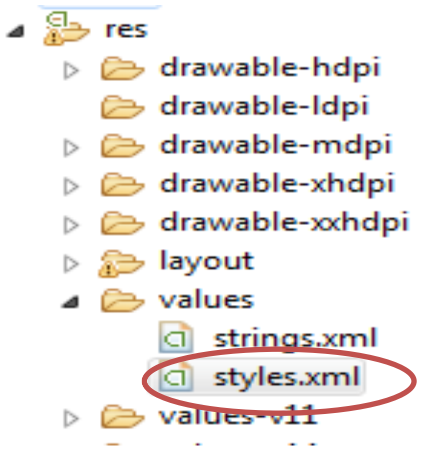
Styles & Themes

Styles & Themes

```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_id"
        style="@style/CustomFontStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onButtonClicked"
        android:text="TestStyle" />
</LinearLayout>
```

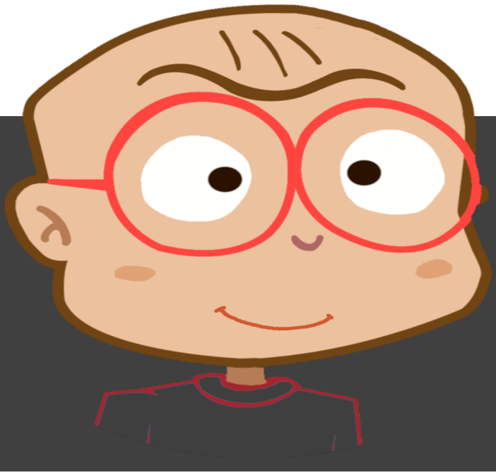
Styles & Themes (cont)



```xml
 <resources>
    <style name="CustomFontStyle">
        <item name="android:capitalize">characters</item>
        <item name="android:typeface">monospace</item>
        <item name="android:textSize">12sp</item>
        <item name="android:textColor">#00FF00</item>
    </style>
    <style name="CustomFontStyle.LargeFont">
        <item name="android:textSize">22sp</item>
    </style>
    <style name="CustomFontStyle.LargeFont.Red">
        <item name="android:textColor">#FF0000</item>
    </style>
</resources>
```

Apply Theme

```
<activity
      android:name="trainingcourse.ActivityXXX"
       android:theme="@style/CustomFontStyle"  >
 </activity>
```

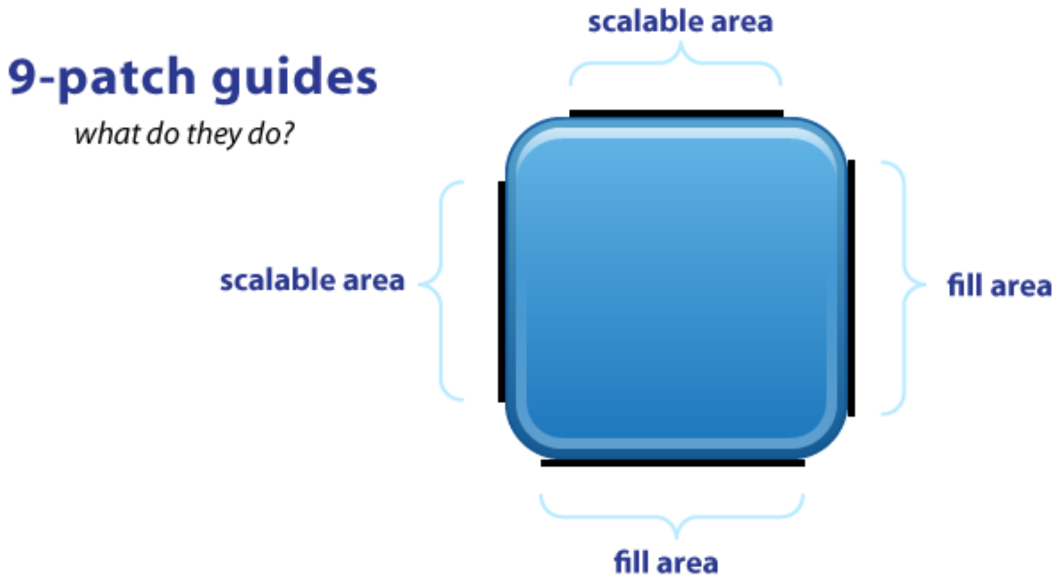


Custom Components

Custom components

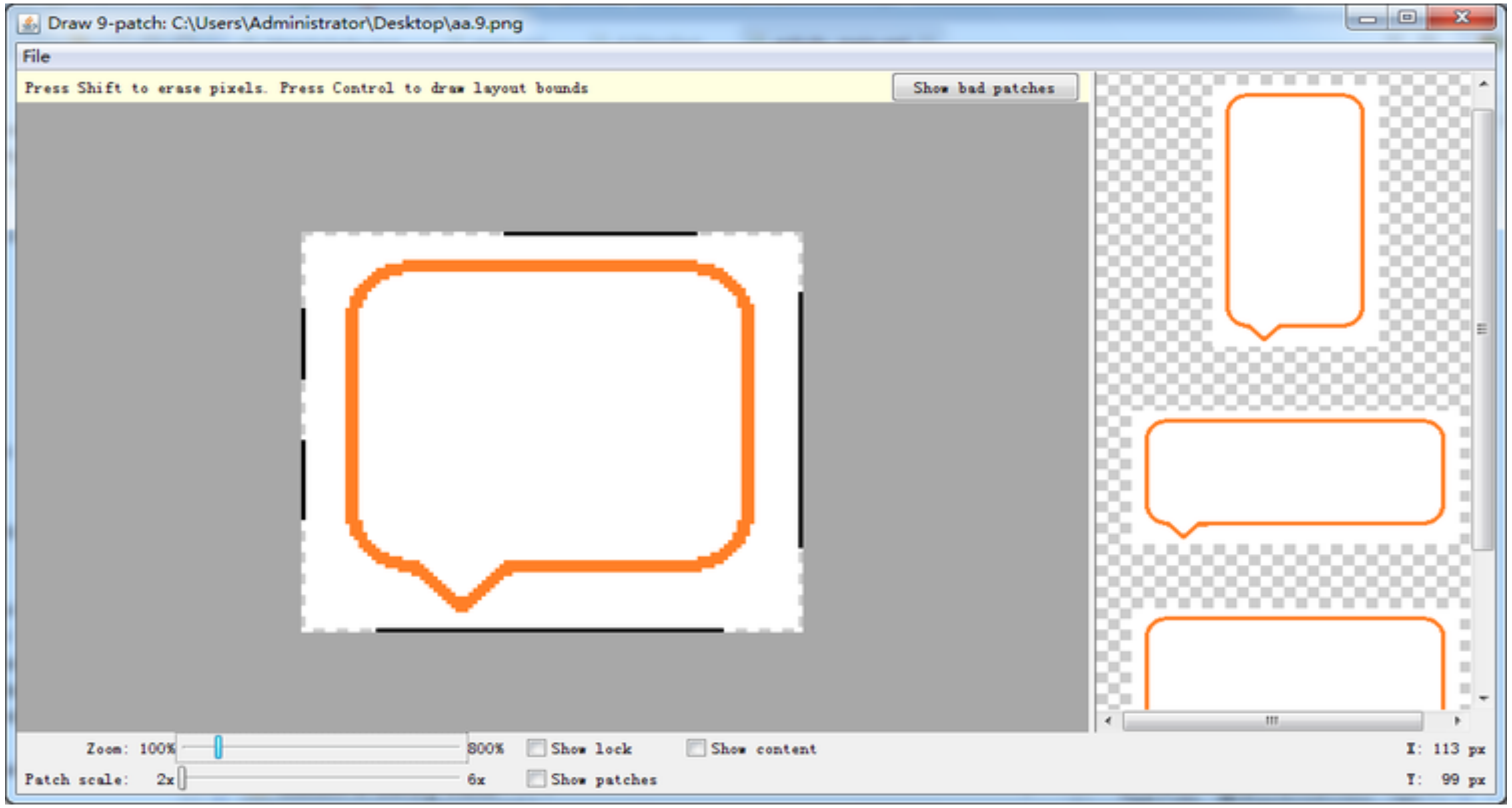
- Draw 9-patch
- Custom Component with Custom Attributes

Custom components



The **Draw 9-patch** tool is a WYSIWYG editor that allows you to create bitmap images that automatically resize to accommodate the contents of the view and the size of the screen. Selected parts of the image are scaled horizontally or vertically based indicators drawn within the image.

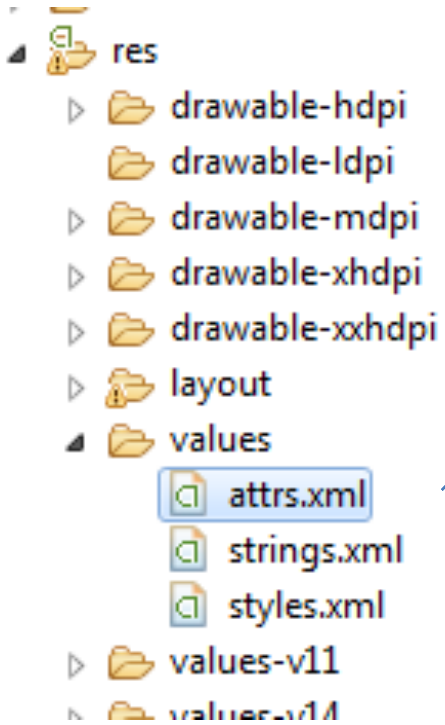
Custom components (cont)


Draw 9-patch: C:\Users\Administrator\Desktop\aa.9.png
File
Press Shift to erase pixels. Press Control to draw layout bounds
Show bad patches
Zoom: 100%
800%
Show lock
Show content
Patch scale: 2x
6x
Show patches
X: 113 px
Y: 99 px

Internal Use

Custom components (cont)

Custom Component with Custom Attribute



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="CustomEditText">
        <attr name="needNumber" format="boolean" />
    </declare-styleable>
</resources>
```

Using

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    .......
     <trainingcourse.androiduserinterface.widget.CustomEditText
            android:id="@+id/customEdittext"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TestStyle"
            app:needNumber="true" />
</LinearLayout>
```

Custom Component with Custom Attribute

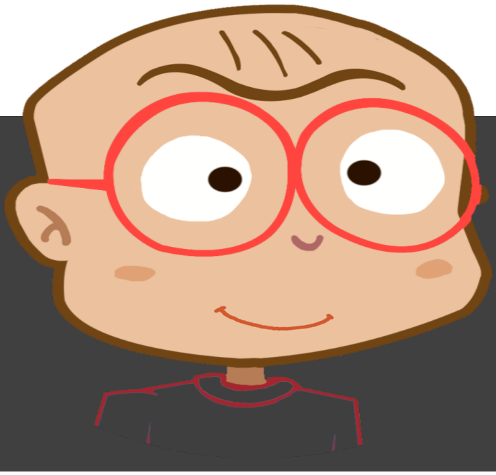
Get Attribute

```
public class CustomEditText extends EditText
{
            ...
TypedArray typedArray = this.context.obtainStyledAttributes(attrs,R.styleable.CustomEditText);
            final int sizeOfType = typedArray.getIndexCount();
            for (int i = 0; i < sizeOfType; ++i) {
                        int attr = typedArray.getIndex(i);
                        switch (attr) {
                                    case R.styleable.CustomEditText_needNumber:
                                    isNeedNumber = typedArray.getBoolean(attr, false);
                                    break;
                        }
            }
            typedArray.recycle();
            ...
}
```

Custom Component with Custom Attribute

Using attribute

```
public void validate() {
        this.setError(null);
        if (isNeedNumber) {
                try {
                        String text = this.getText().toString();
                        text = text.replaceAll("\\D", "");
                        if (text.length() == 0) {
                        this.setError("Please check data input!!!");
                }
                } catch (Exception e) {
                }
        }
}
```



Shape & Selector

Shape

Shape Drawable
This is a generic shape defined in XML.

- FILE LOCATION:res/drawable/*filename*.xml
- RESOURCE REFERENCE:In Java: R.drawable.*filename*
- In XML: @[*package*:]drawable/*filename*

Shape & Selector (cont)

Shape

```
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="#1e1e1e" />
    <stroke
        android:width="1dp"
        android:color="#55ffffff" />
    <corners
        android:bottomLeftRadius="8dp"
        android:bottomRightRadius="8dp"
        android:topLeftRadius="8dp"
        android:topRightRadius="8dp" />
    <gradient
        android:angle="0"
        android:endColor="#ef5429"
        android:startColor="#ad3210" />
</shape>
```

Shape & Selector (cont)

Selector

- FILE LOCATION:res/color/*filename*.xml
- RESOURCE REFERENCE:In Java: R.color.*filename*
- In XML: @[*package*:]color/*filename*

Shape & Selector (cont)

Selector

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/bg_blue_btn1_p"
android:state_pressed="true"/>
    <item android:drawable="@drawable/bg_blue_btn1_p"
android:state_focused="true"/>
    <item android:drawable="@drawable/bg_blue_btn1"/>

</selector>
```

Exit Course

THANK YOU