



THIẾT KẾ GIAO DIỆN ANDROID

BÀI 2: WIDGET, THIẾT KẾ WIDGET

- ⊙ Kết thúc bài học này bạn có khả năng
 - ⊙ Hiểu được các loại widget (view)thông dụng
 - ⊙ Biết được cách chọn lựa view phù hợp
 - ⊙ Biết cách thay đổi thiết kế cho các view



Phần I: Giới thiệu về Android UI

 A Giới thiệu các view cơ bản

 B Cây kế thừa view


 C TextView, Button, EditText,.....

Phần II: Chỉnh giao diện cho các view

 D Style

 E Theme

 F Selector

 G 9-patch





BÀI 2: WIDGET, THIẾT KẾ WIDGET

PHẦN 1: WIDGET CƠ BẢN

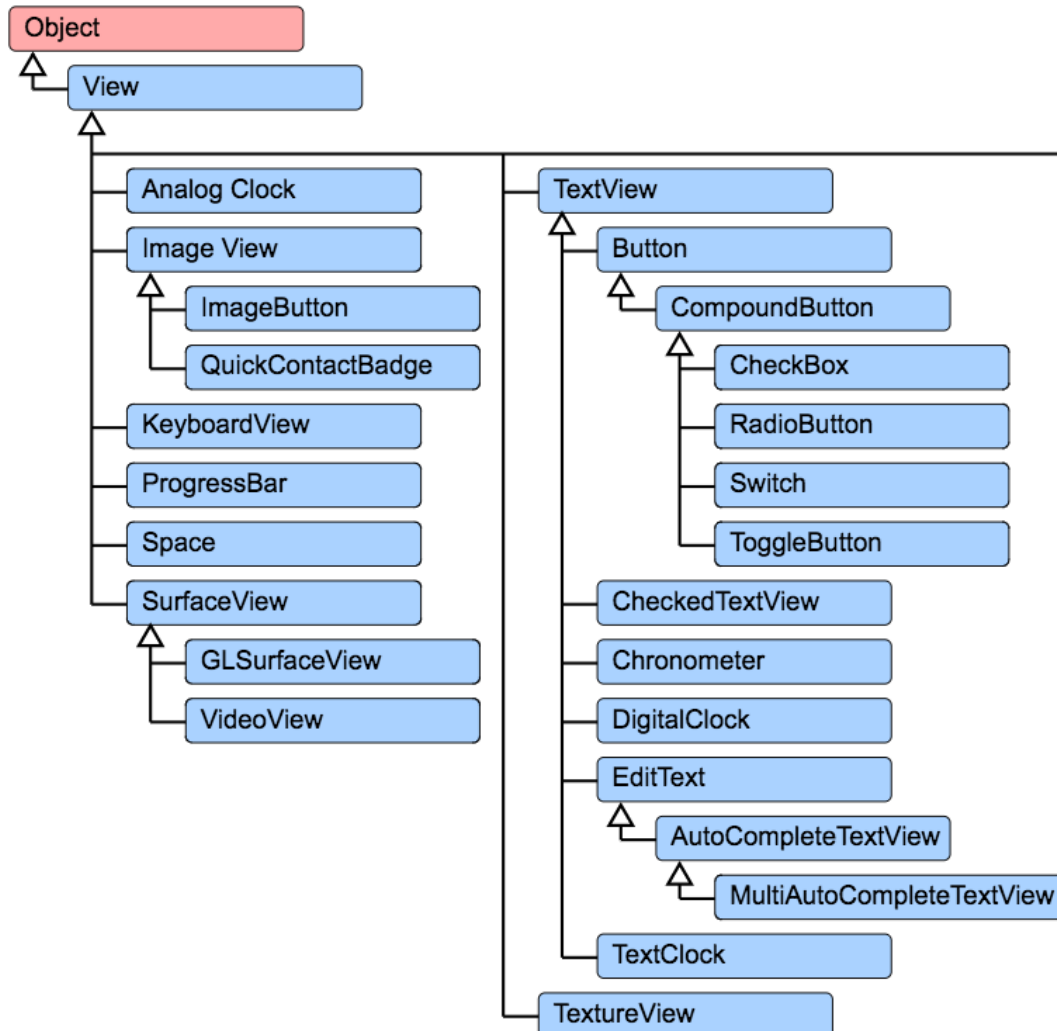
- ❑ Widget hay View là các đối tượng hiển thị trên giao diện đồ họa.
- ❑ Muốn dùng nó trong file java ta phải ánh xạ nó bằng lệnh `findViewById()` trong file java với đối số truyền vào là id trên file layout

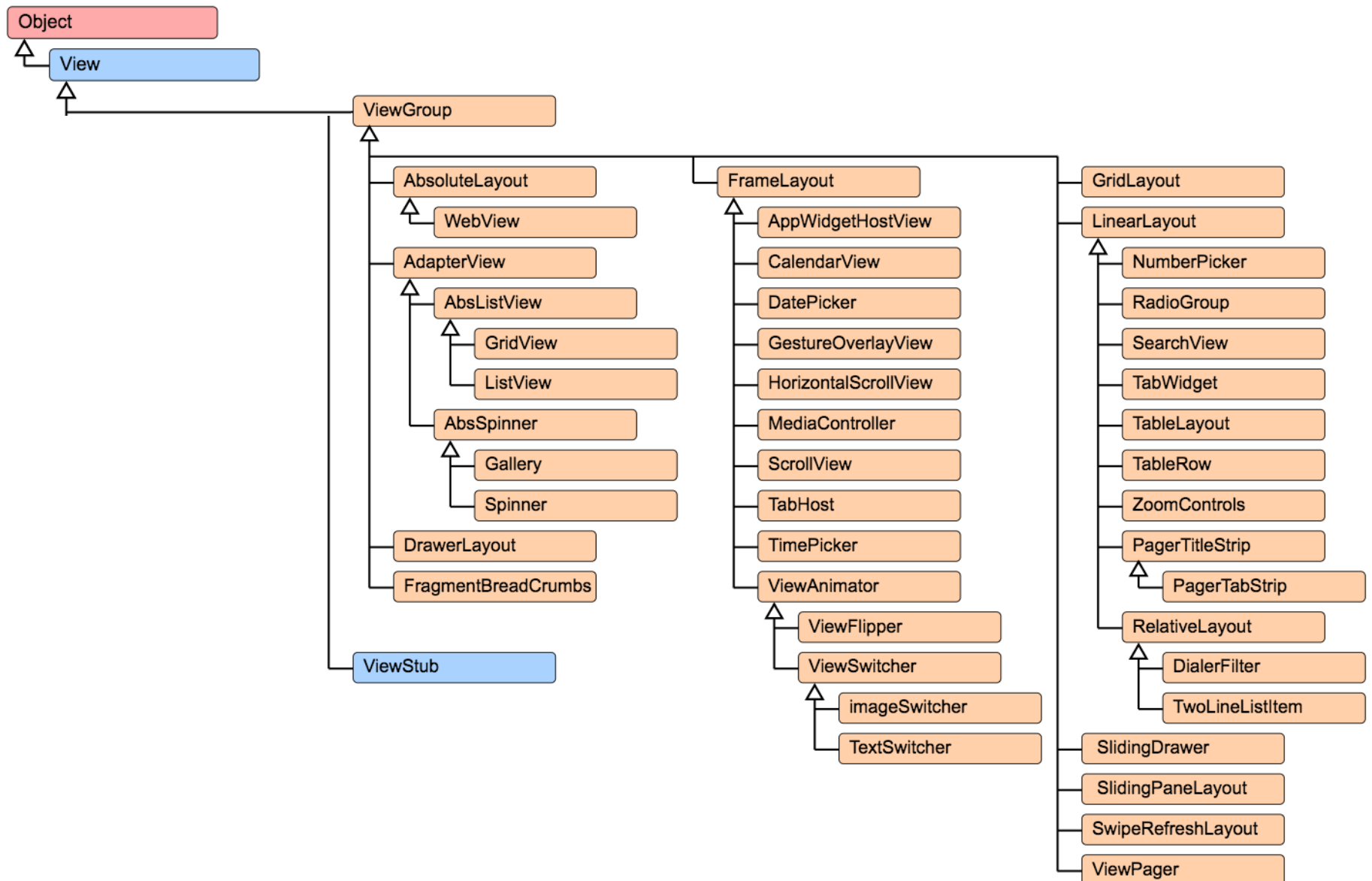
<Button

```
android:id="@+id/button1"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_weight="1"  
android:text="Button" />
```

```
public class MainActivity extends AppCompatActivity {  
    Button bt;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        bt=(Button)findViewById(R.id.button1);  
    }  
}
```

- Tùy theo sự kế thừa mà một view sẽ có các thuộc tính, phương thức, sự kiện chung hoặc riêng





- ❑ Tra cứu các cây kế thừa của view, các thuộc tính, phương thức, sự kiện... theo địa chỉ <https://developer.android.com/reference/android/view/View.html>
- ❑ Các thuộc tính thông thường bắt đầu bằng android:
- ❑ Thuộc tính cần lưu ý:
 - ❖ id: dùng để ánh xạ nó trong file java
 - ❖ Layout_width, layout_height: rộng, cao của view. Dùng wrap_content, match_parent, hoặc số đo (nên dùng đơn vị dp)

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button" />
```


❑ Hiển thị 1 chuỗi lên giao diện

❑ Lưu ý:

❖ Gán chuỗi trong layout:

`android:text="hello world"`

❖ Gán chuỗi bằng java:

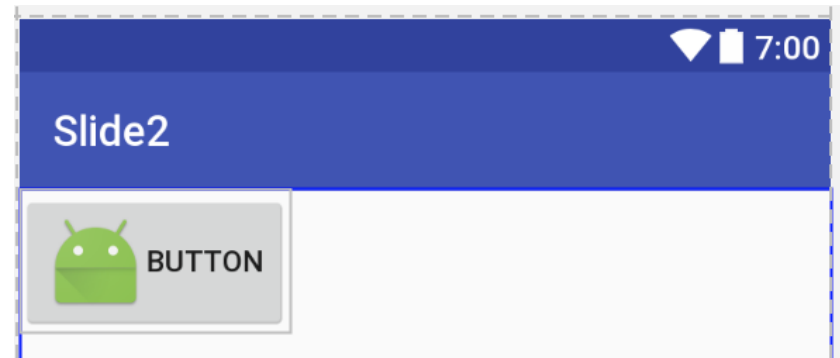
`txt1.setText("Hello world");`

❖ Lấy chuỗi trong:

`String msg=txt1.getText().toString();`

- ❑ Nút nhấn.
- ❑ Có thể chứa chuỗi, icon, hoặc vừa chuỗi vừa icon

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:drawableLeft="@mipmap/ic_launcher"  
    android:text="Button" />
```



- ❑ Bắt sự kiện: trong java ánh xạ và bắt sự kiện. Sự kiện thông dụng nhất là onClickListener

```
public class MainActivity extends AppCompatActivity {  
    Button bt;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        bt=(Button)findViewById(R.id.button1);  
        bt.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                  
            }  
        });  
    }  
}
```

- ❑ Cho phép người dùng nhập dữ liệu vào
- ❑ Lấy dữ liệu trong java (lưu ý getText chỉ trả về Editable, phải toString mới trả về được chuỗi)
String x = et1.getText().toString();
- ❑ Gán dữ liệu trong java (lưu ý setText chỉ nhận vào chuỗi):
et1.setText("hello");

- ❑ Thuộc tính inputType quy định kiểu hiển thị (vd: password) hoặc kiểu hiển thị bàn phím

```
<EditText  
    android:id="@+id/editText3"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="" />
```

LinearLayout>

date
datetime
none
number
numberDecimal
numberPassword
numberSigned
phone
text
textAutoComplete
textAutoCorrect

- ❑ Dùng hiển thị hình ảnh.
- ❑ Trong file layout có thể lấy hình trong thư mục drawable hoặc mipmap.
- ❑ Version mới thay vì dùng android:src có thể dùng app:srcCompat để chỉ đến hình

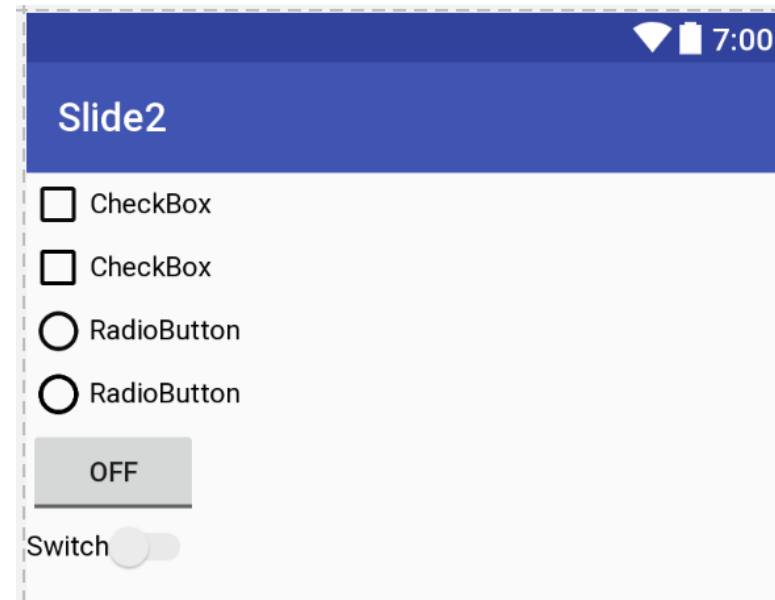
```
<ImageView  
    android:id="@+id/imageView"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:srcCompat="@mipmap/ic_launcher" />
```

- ❑ Trong java chỉ đến hình dùng:

`Iv.setImageResource(R.mipmap.hinh);`

CHECKBOX, RADIOBUTTON, TOGGLEBUTTON, SWITCH

- ❑ 4 view này xử lý khá giống nhau đều có 2 trạng thái checked và unchecked
- ❑ Checkbox cho phép chọn nhiều trong nhiều
- ❑ RadioButton cho phép chọn 1 trong nhiều (phải bỏ vào một RadioGroup)
- ❑ ToggleButton và Switch chỉ khác nhau về hình dạng (switch đến api 11 mới có). Có thêm 2 thuộc tính là textOn và textOff cho text ở 2 trạng thái
- ❑ Gán trạng thái mặc định trong layout dùng:
android:checked="true"



CHECKBOX, RADIOBUTTON, TOGGLEBUTTON, SWITCH

❑ Xét trạng thái hiện tại:

```
if (cb.isChecked() == true)
{
}
```

❑ Gán trạng thái hiện

```
cb.setChecked(true);
```

❑ Đảo trạng thái hiện
tại `cb.toggle();`

❑ Bắt sự kiện khi có sự thay đổi trạng

```
cb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
    }
});|
```

❑ Xét trạng thái theo RadioGroup:

```
RadioGroup group= (RadioGroup) findViewById(R.id.radiogroup1);
int idChecked = group.getCheckedRadioButtonId();
switch (idChecked)
{
    case R.id.radioButton: break;
}
```



DEMO

Các view cơ bản



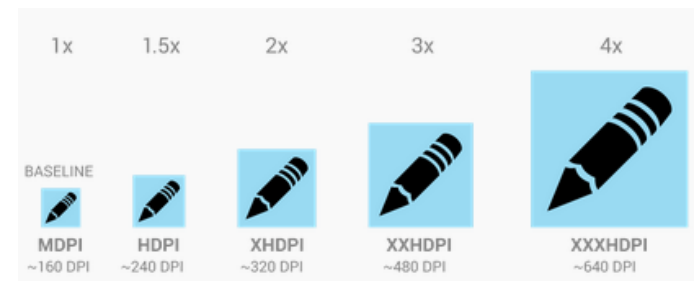
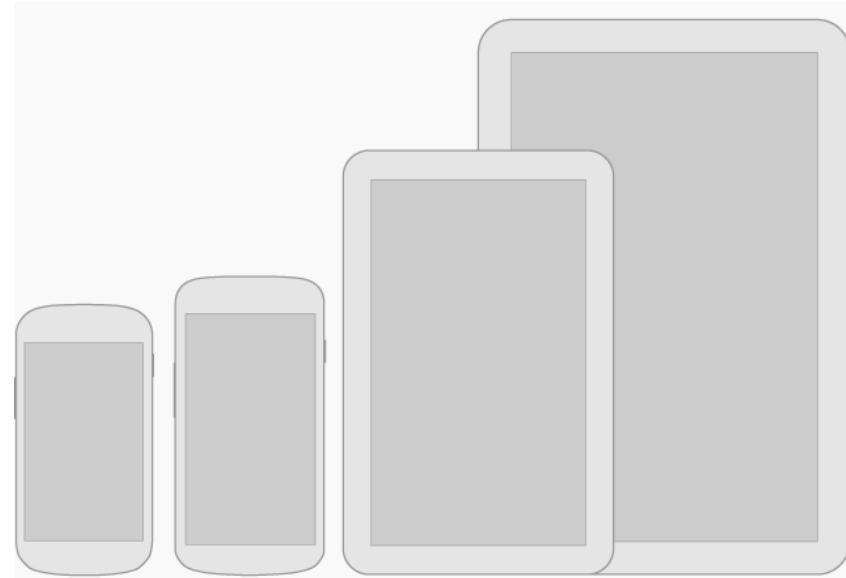


BÀI 2: WIDGET, THIẾT KẾ WIDGET

PHẦN 2: STYLE, THEME

❑ **Devices và Displays**

- Android là nền tảng hệ điều hành được sử dụng trên rất nhiều thiết bị lớn nhỏ khác nhau như: tablet, phone.
- Khi thiết kế cần phải đảm bảo flexible (linh hoạt): dẫn hoặc nén layout tùy theo độ rộng và cao của màn hình, vì vậy xu hướng thiết kế hiện nay là multi-pane layout (layout đáp ứng cho nhiều kiểu giao diện)



Color

- ❖ Chọn màu sắc phù hợp với ứng dụng sẽ tăng hiệu quả sử dụng. Chú ý trong thiết kế vì người mù màu thường không phân biệt được màu đỏ và xanh lá cây.



Palettte

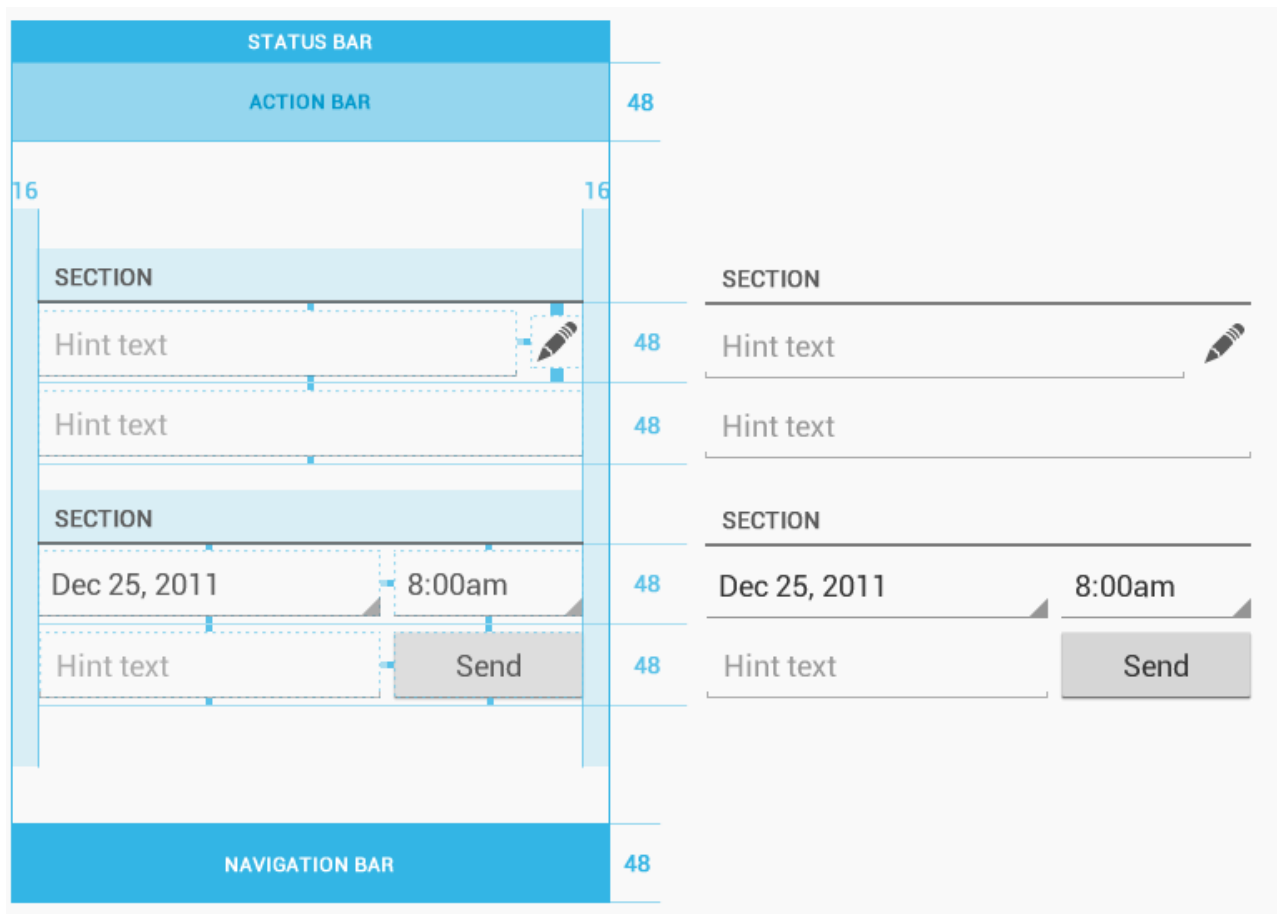
- ❖ Mỗi một màu sẽ có một dải màu từ tối đến sáng cho người thiết kế lựa chọn





□ Metrics và Grids

- Một ví dụ về thiết kế bố trí màn hình:



- ❑ Style là các thuộc tính của view được tách riêng thành style và áp trở lại cho nhiều view (giống như CSS của web).
- ❑ Một style là một bộ các attribute/value được khai báo sẵn để apply vào look & feel (một các nói khác về GUI) của view
- ❑ Style được tạo trong file styles.xml trong thư mục res/values.
- Style có thể được kế thừa bằng cách thêm thuộc tính parent = "@android: style/..."

- ❑ Ví dụ 1 view có các thuộc tính như hình:

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#ff0000"
    android:text="Button" />
```

- ❑ Tách các thuộc tính ra bỏ vào style:

```
<style name="nut">
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textColor">#ff0000</item>
</style>
```

- ❑ Sau đó có thể áp vào view như sau:

```
<Button
    style="@style/nut"
    android:id="@+id/button2"
    android:text="Button" />
```

- ❑ Một style sau đó có thể áp vào nhiều view

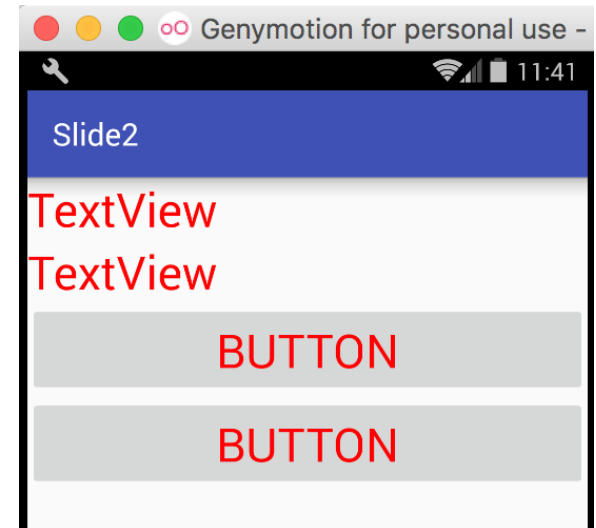
- ❑ Theme là một style mà nó apply cho toàn bộ activity hoặc thậm chí toàn application
- ❑ Thêm thuộc tính `android:theme` vào activity trong manifest
- ❑ Khi một style được apply thành theme thì toàn bộ các thuộc tính được khai báo trong style sẽ ghi đè giá trị mặc định của các view trong Activity

❑ Trong file style.xml

```
<style name="MyTheme" parent="AppTheme">  
    <item name="android:textSize">30dp</item>  
    <item name="android:textColor">#ff0000</item>  
</style>
```

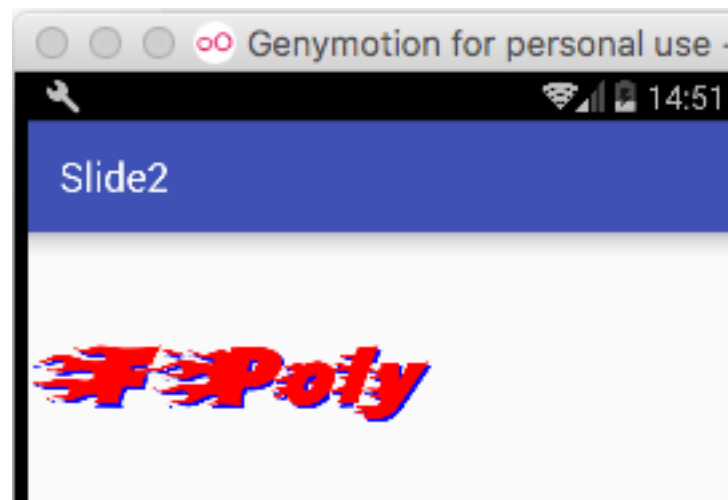
❑ Trong file AndroidManifest.xml

```
<activity android:name=".Main2Activity"  
    android:theme="@style/MyTheme">  
</activity>
```



- ❑ Tạo thư mục assets (nếu chưa có, chép font vào thư mục assets)
- ❑ Trong code java:

```
TextView tv1=(TextView)findViewById(R.id.textView1);  
Typeface font=Typeface.createFromAsset(getAssets(),"Blazed.ttf");  
tv1.setTypeface(font);
```

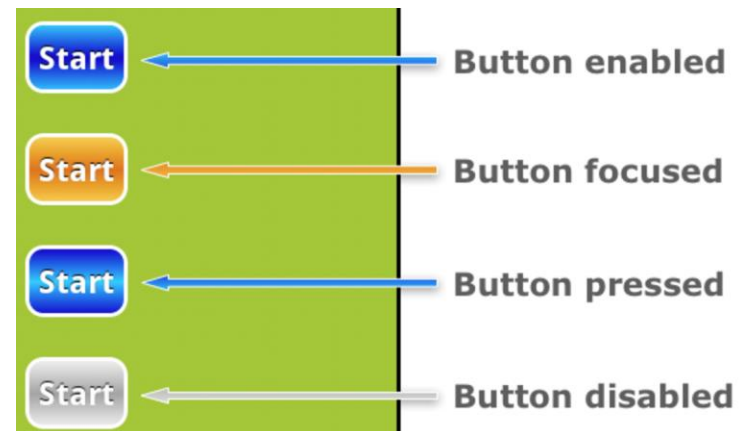


- Giúp tạo background cho các control có 2 hoặc nhiều trạng thái vd: bình thường & được nhấn, checked & unchecked...

ListRow Default State



ListRow Hover State



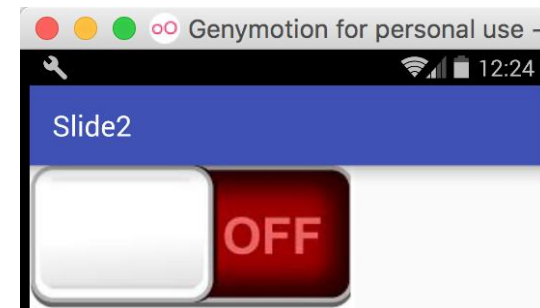
Trong thư mục drawable tạo file mới dạng Drawable resource file.

Tạo ra 2 cặp thẻ item cho 2 trạng thái (1 checked, 1

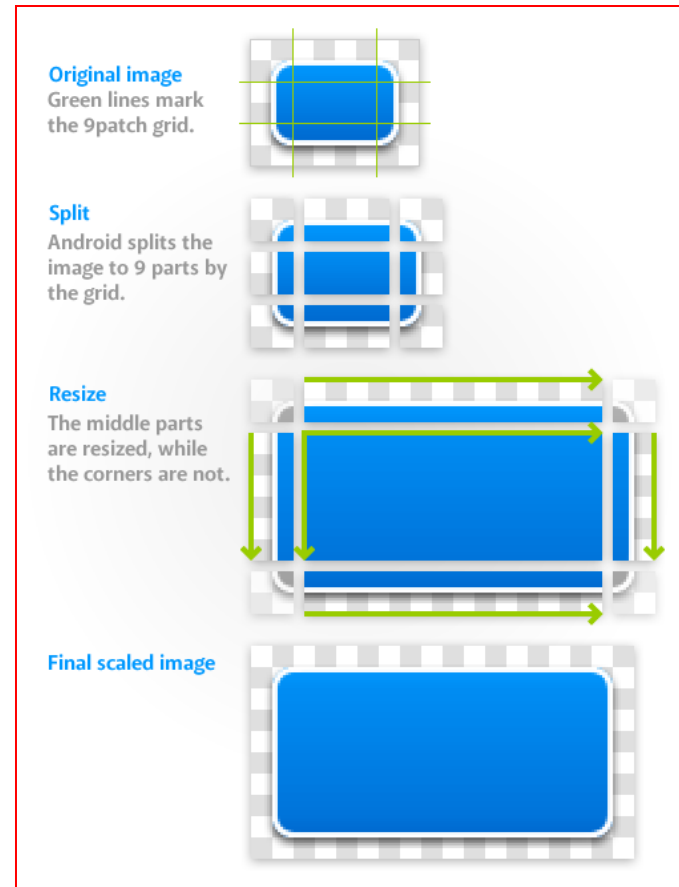
```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/toggleon" android:state_checked="true" />
    <item android:drawable="@drawable/toggleoff" android:state_checked="false" />
</selector>
```

Gán background vào View

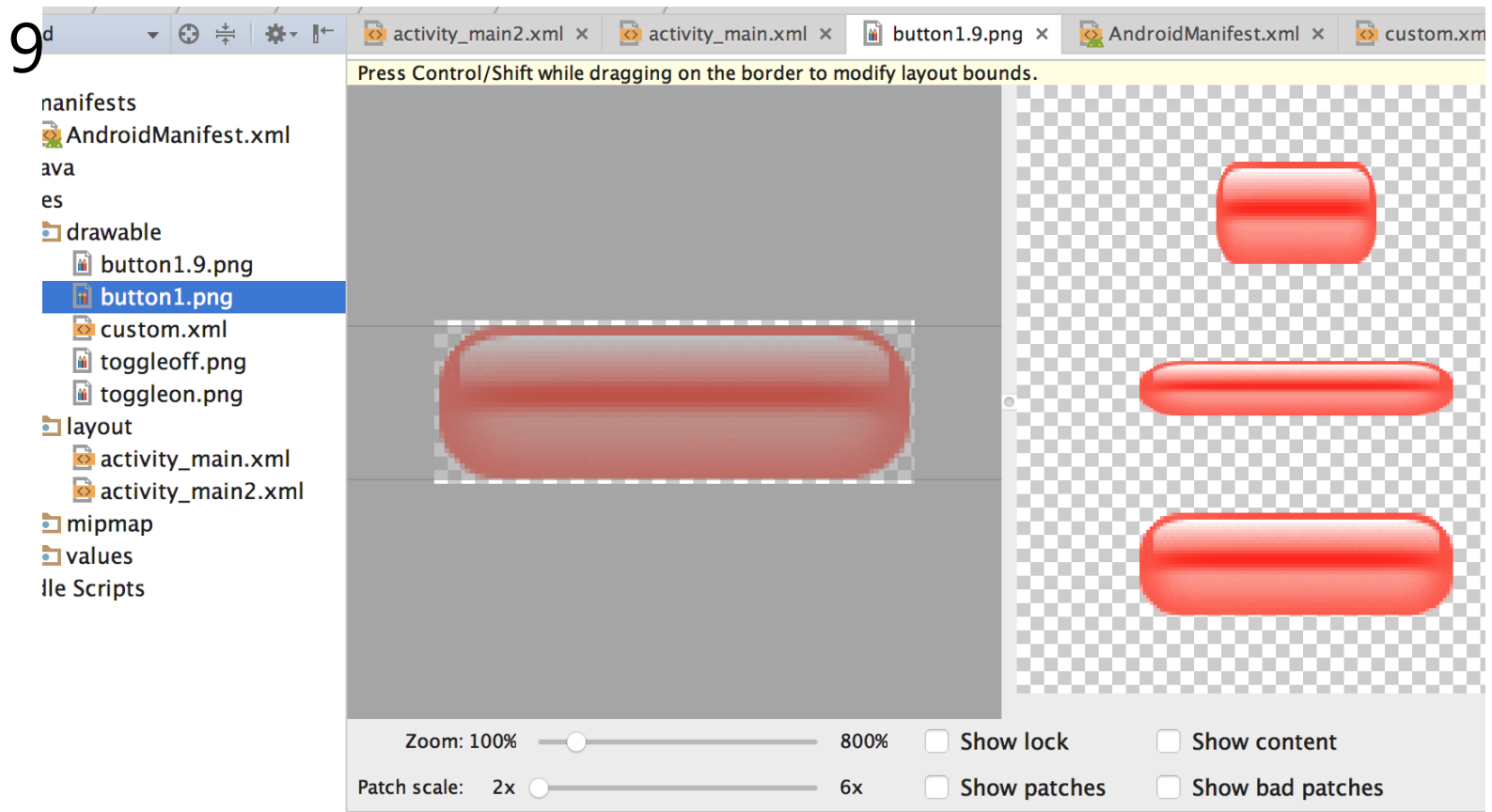
```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/toggleButton1"
    android:textOn=""
    android:textOff=""
    android:background="@drawable/custom"
/>
```



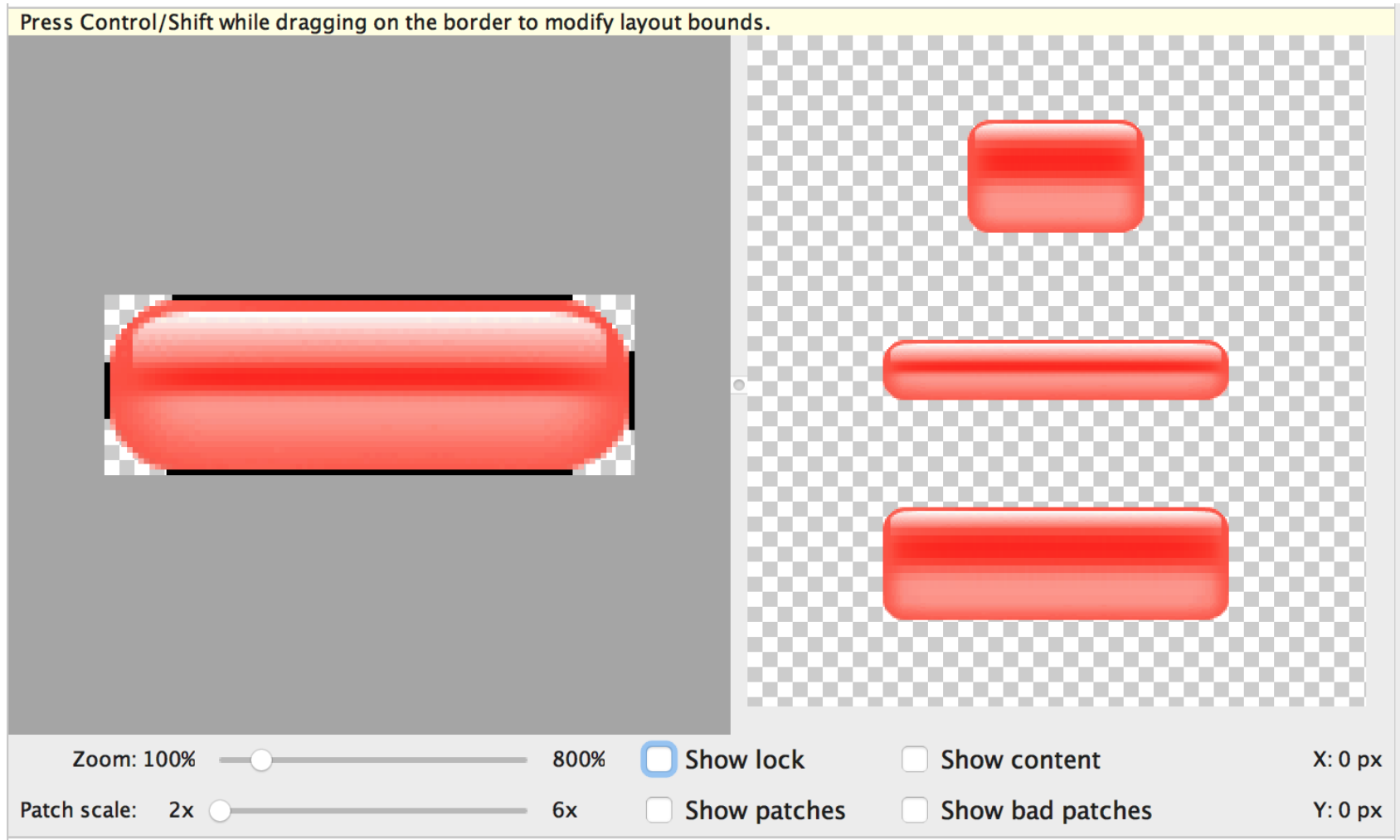
- ❑ Hình 9-patch là hình tự động thay đổi kích thước để phù hợp với nội dung của view và kích thước màn hình.
- ❑ Các phần lựa chọn của hình ảnh sẽ được co giãn theo chiều dọc, ngang hoặc không co giãn dựa theo những quy định do ta vẽ ra



- Trong thư mục drawable, click phải hình cần tạo 9-patch chọn Create 9-patch file để tạo file hình mới. Double click hình mới để mở công cụ Draw



□ Vẽ như hình và xem kết quả ở phần bên phải





DEMO

Style
Theme

Selector
9-patch



- ☐ Các widget cơ bản
- ☐ Style
- ☐ Theme
- ☐ Selector
- ☐ 9-patch





Cảm ơn