CV4

build.gradle

Struktura build.gradle

- `compileSdkVersion` (25)
- `buildToolsVersion` (25.0.2)
- `applicationId` (com.application.my)
- `minSdkVersion` (16)
- `targetSdkVersion` (25)
- `versionCode` (1234)
- `versionName` (1.2.3-AHOJ)
- `dependencies` (jCenter, mavenCentral, JAR)

Build types

https://developer.android.com/studio/build/build-variants.html

- debug, release + vlastní
- dokážeme přizpůsobovat parametry při vytváření APK
  - SigningConfig
  - Shrinking code (`minifyEnabled true`)
  - Proguard (`proguardFiles getDefaultProguardFile`)

Build types

```
buildTypes {
        debug {
            minifyEnabled false
            shrinkResources false
            signingConfig signingConfigs.debug
        }
        client {
            …
        }
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile(‘proguard-android.txt')
            signingConfig signingConfigs.release
        }
}
```

ProductFlavors

- varianty aplikace s různou “příchutí”
- různé `res` a/nebo Java kód pro každou variantu
- demo/full, test/live …

```
productFlavors {
        demo {
            applicationIdSuffix ".demo"
            versionNameSuffix "-demo"
        }
        full {
            applicationIdSuffix ".full"
            versionNameSuffix "-full"
        }
    }
```

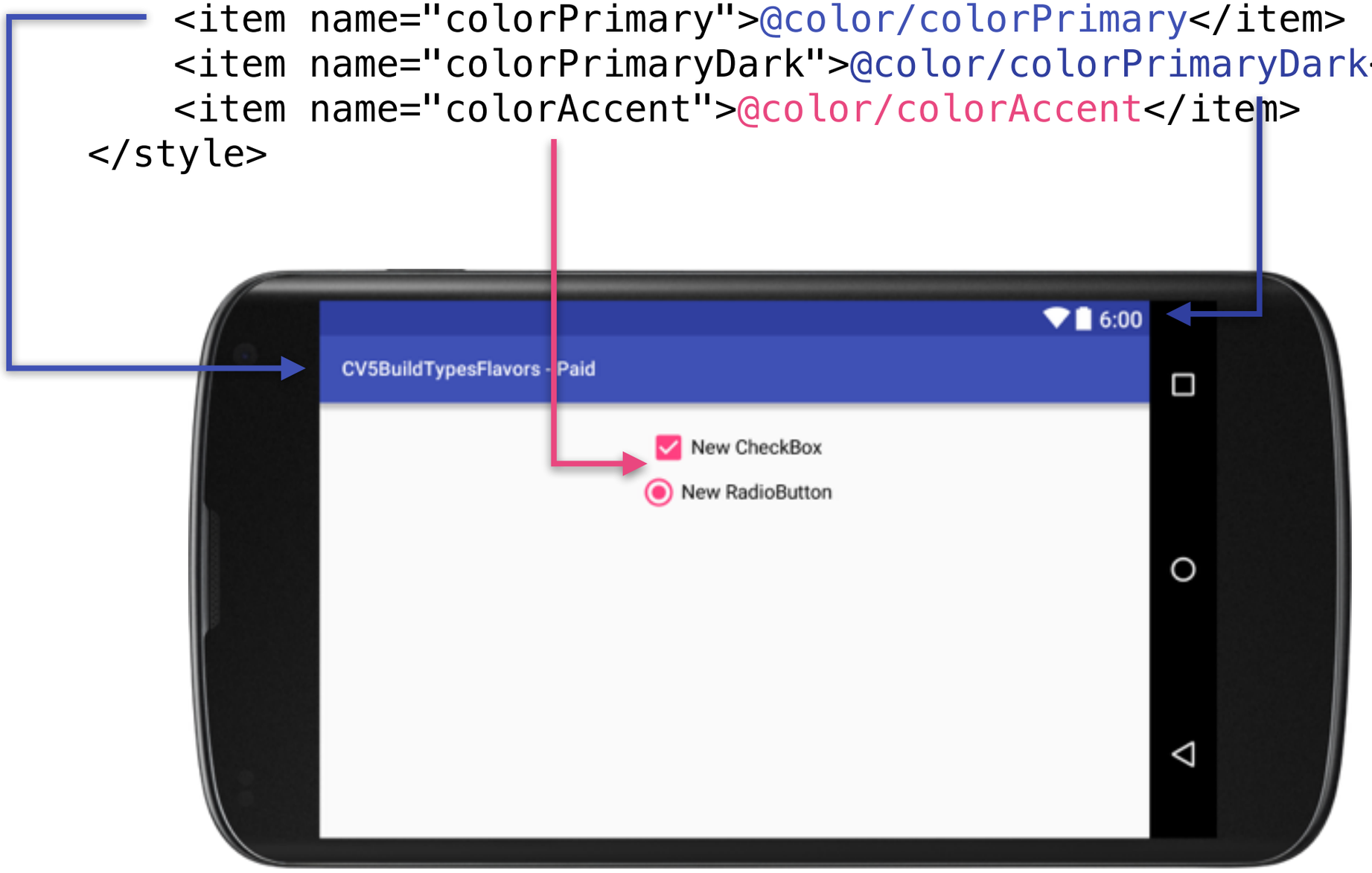
Styly a témata

- **style** - sada XML atributů ovlivňujících vzhled UI
  - velikosti, barvy, stíny, písma, …
- **theme** - style aplikovaný na celou Activity
  - ovlivňuje všechny View, které nastavený XML atribut podporují
  - může být přiřazen i k View
- zajišťují konzistenci UI, urychlují vývoj

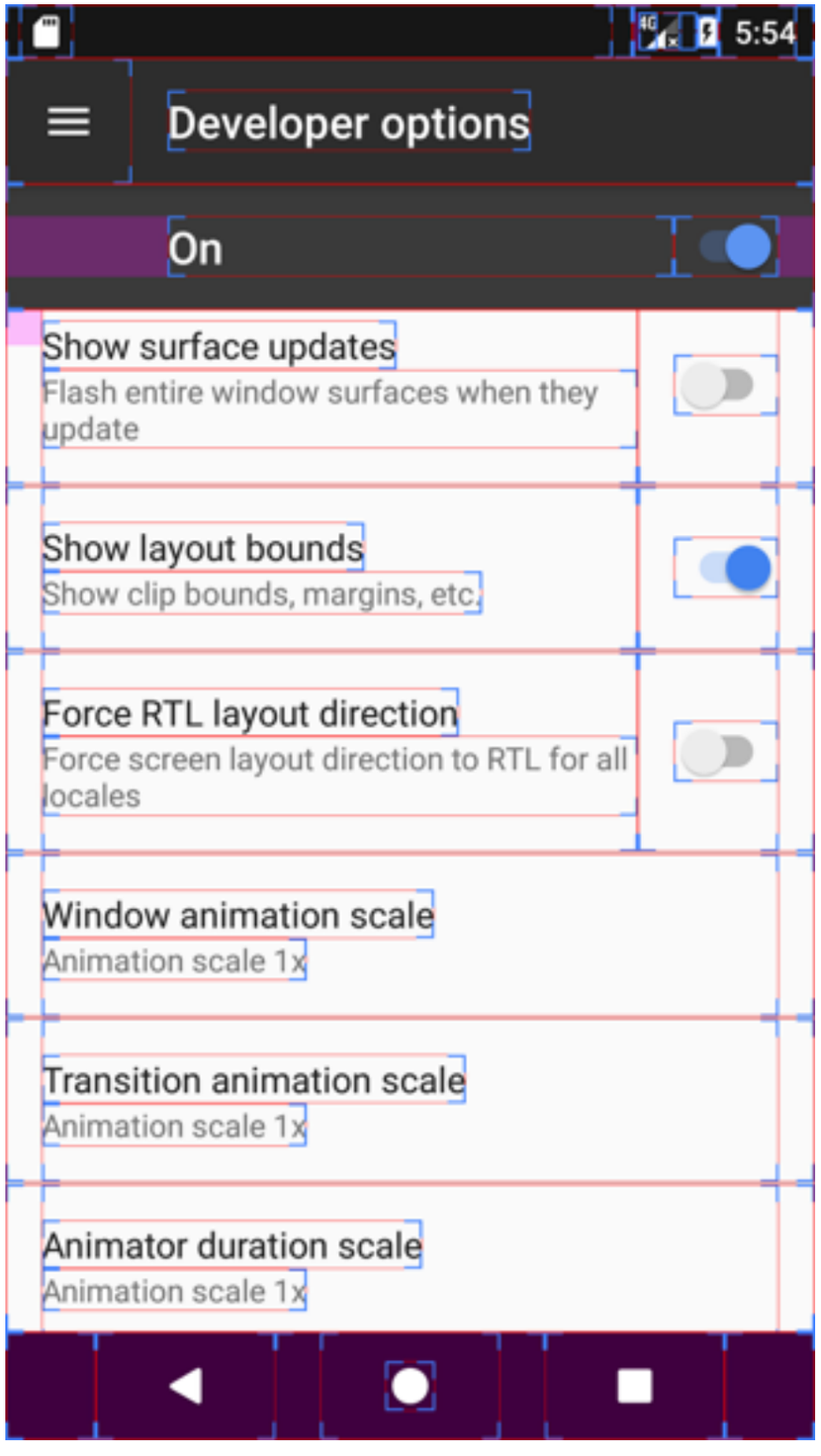
Témata

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
```



Styly



- color
- drawable
- selector
- shape

Styles

```
<style name="MyButton" parent="Button">
    <item name=“android:textColor”>@color/red</item>
</style>
```

```
<Button
    style=“@style/MyButton”
    … />
```

Selector

- Změna atributů View na základě změny stavu
  - `pressed`, `checked`, `selected` apod.
- drawable, string, color,…
- ale! -> `View` musí změnu stavu podporovat

Selector

color/textview_color.xml

```
<selector>
    <item
        android:color="@color/colorAccent"
        android:state_pressed="true"/>
    <item
        android:color="@color/colorPrimary" />
</selector>
```

…

```
android:textColor=“@color/textview_color”
```

…

Shape

- Definice pomocí XML

- Shape, Bitmap, Gradient apod.

Shape

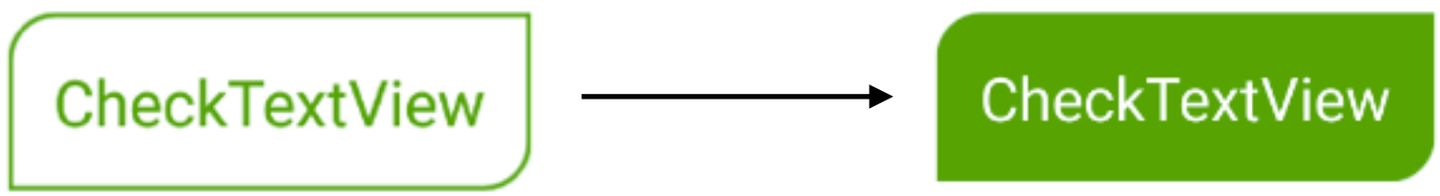
```
<shape>
    <solid
      android:color="@color/colorPrimary"/>
    <corners
        android:radius="10dp" />
</shape>
```

<layer-list />

…
android:textColor=“@drawable/bg”
…

Zadání

- Vytvořit CheckTextView:



Řešení

styles.xml

```
<style name=“checkedViewStyle">
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:padding">10dp</item>
    <item name=“android:textColor”>@color/text</item>
    <item name=“android:background”>@drawable/bg</item>
</style>
```

color/text.xml

```
<selector>
    <item android:color="@android:color/white" android:state_checked="true" />
    <item android:color="@android:color/green" />
</selector>
```

drawable/bg.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/bg_checked" android:state_checked="true" />
    <item android:drawable="@drawable/bg_default"  />
</selector>
```

drawable/bg_default.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@android:color/white" />
    <stroke android:width="2dp" android:color="@android:color/holo_green_dark" />
    <corners android:topLeftRadius="20dp" android:bottomRightRadius="20dp" />
</shape>
```

drawable/bg_checked.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@android:color/holo_green_dark" />
    <corners android:topLeftRadius="20dp" android:bottomRightRadius="20dp" />
</shape>
```

Animace

- rozpohybování UI
- Java i XML
- musíme si je psát sami
- interpolatory
- property animator, view animator

Animace - XML

android:animateLayoutChanges="true"

```
AnimationUtils.loadAnimation(
        context,
        R.anim.my_animation);
```

Animace - Java

```
ValueAnimator animation = ValueAnimator.ofInt(1, 10);
animation.setDuration(5000);
animation.start();
animation.addUpdateListener(new
ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        t.setText(String.valueOf(animation.getAnimatedValue()));
    }
});


ObjectAnimator anim =
   ObjectAnimator.ofFloat(t, "textSize", 10, 30);
anim.setDuration(5000);
anim.start();
```

Animace - Java

- AnimatorListener
- AnimatorUpdateListener
- AnimatorSet
- Interpolator

Zadání

- Vytvořte aplikaci, která bude bude zobrazovat číslo od 1 do 1000 během 5 vteřin. Zároveň se bude měnit velikost písma z 10 na 30 bodů po dobu 5 vteřin.

Řešení

```
ValueAnimator animation = ValueAnimator.ofInt(1, 1000);
animation.setDuration(5000);
animation.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator animation) {
        t.setText(String.valueOf(animation.getAnimatedValue()));
    }
});

ObjectAnimator animator2 = ObjectAnimator.ofFloat(t, "textSize", 10, 30);
animator2.setDuration(5000);

AnimatorSet animatorSet = new AnimatorSet();
animatorSet.play(animation).with(animator2);
animatorSet.start();
```