

# Android

## Felhasználói felület, Fragmentek

Dr. Ekler Péter

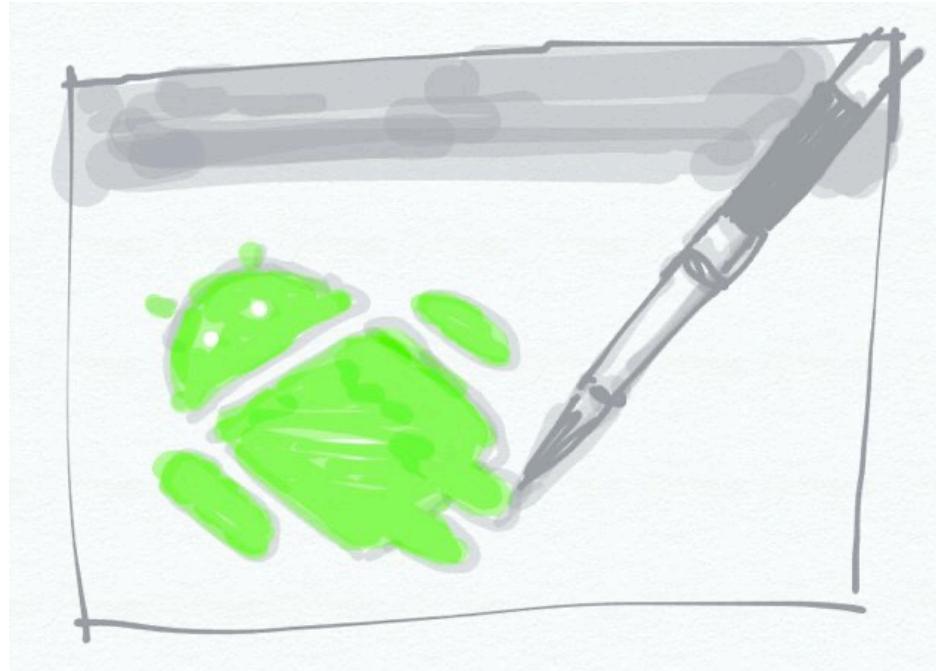
[peter.ekler@aut.bme.hu](mailto:peter.ekler@aut.bme.hu)



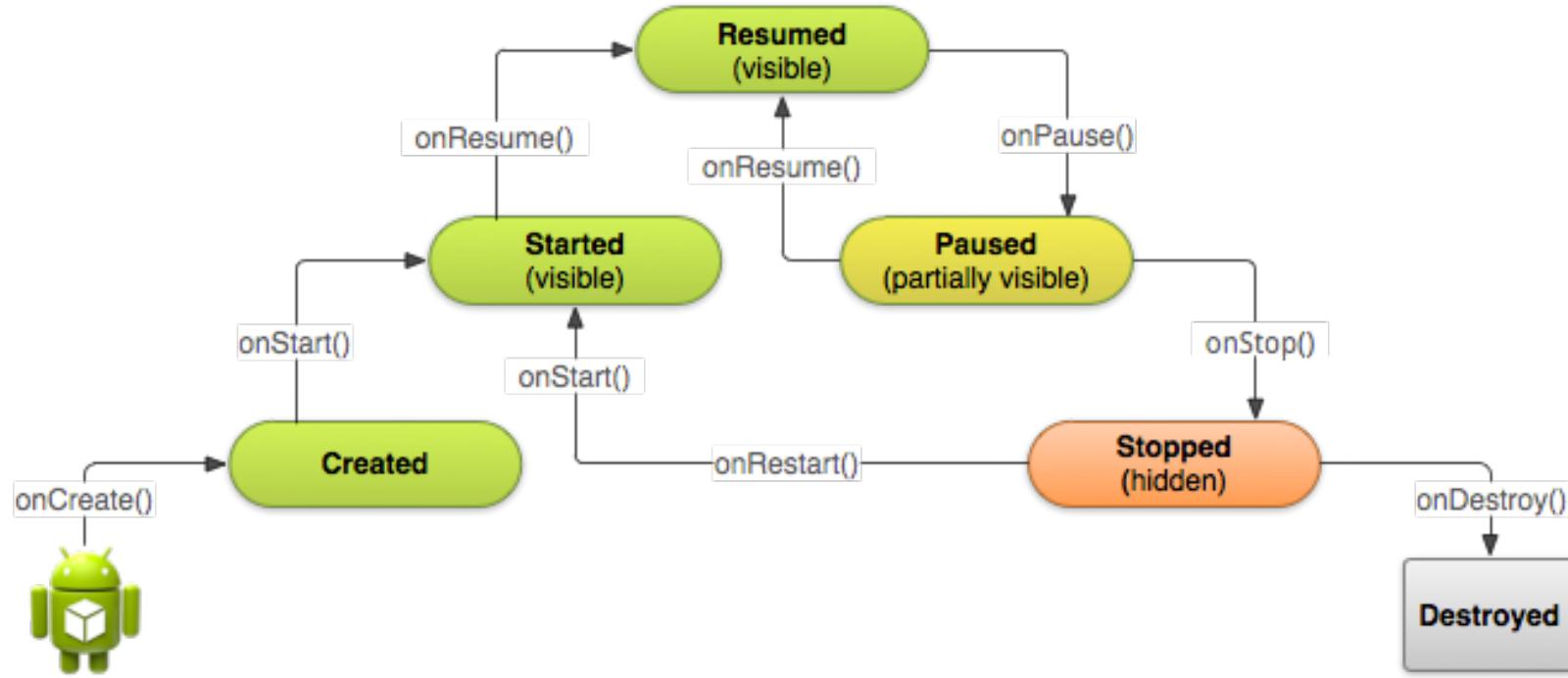
Department of  
Automation and  
Applied Informatics

# Activity életciklus

- Rajzoljuk le az Activity életciklust!



# Activity életciklus



# Hogy is volt?

- Egy Android alkalmazás milyen komponensekből épülhet fel?
- Mi a Service komponens?
- Miket kell tartalmaznia a manifest állománynak?
- Az Activity callback életciklus-függvények felüldefiniálásakor meg kell-e hívni kötelezően az ős osztály implementációját?
- Ha A Activity-ből átváltunk B Activity-re, milyen sorrendben hívódnak meg az életciklus függvények?
- Magyarázza el az Activity Back Stack működési elvét!

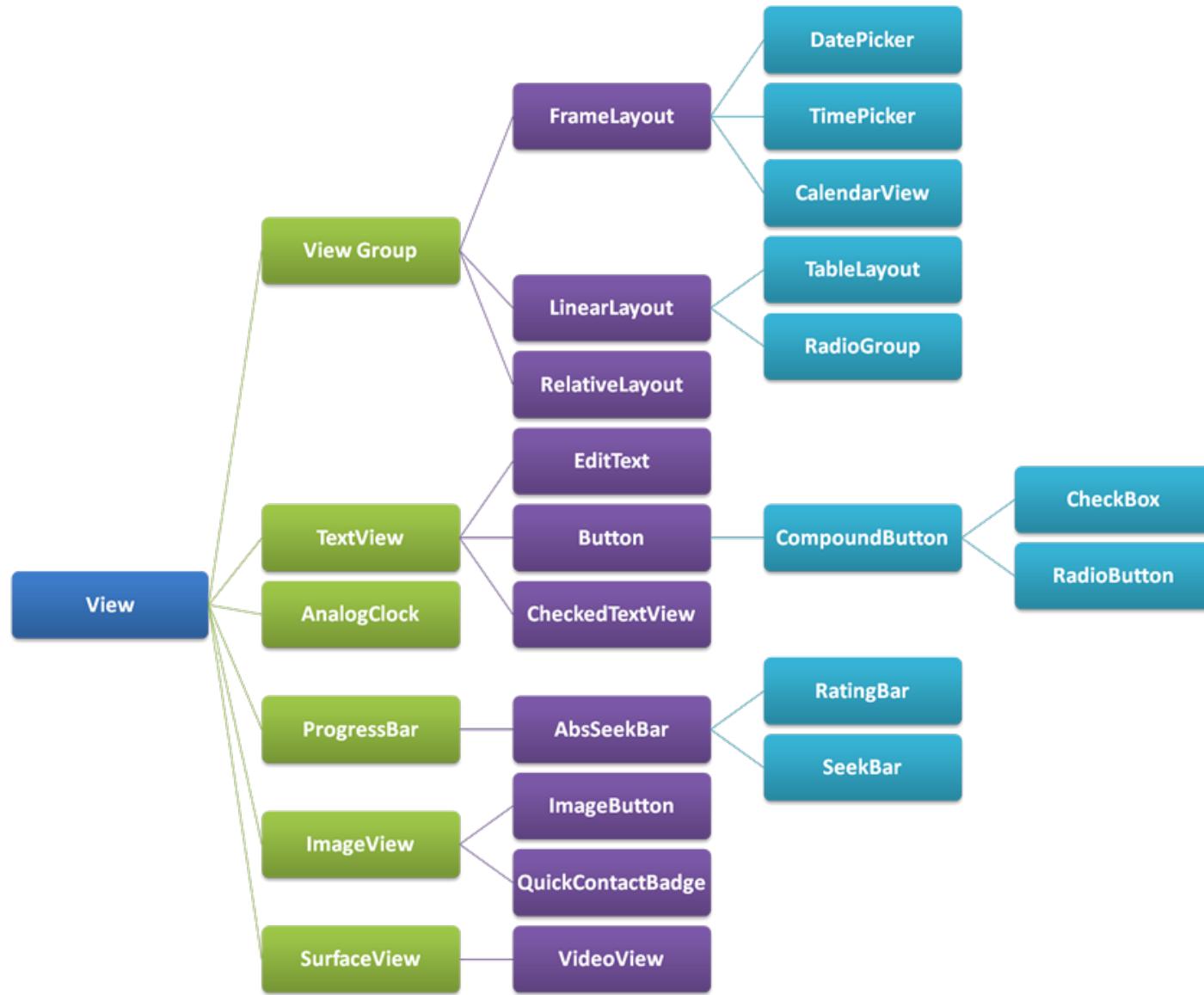
# Hogy is volt?

- Mit értünk a sűrűségfüggetlen pixel fogalom alatt?
- Egy 320 dpi-s képernyőn, 1 dp mennyi fizikai pixelnek felel meg ?
- Vázolja fel egy Android alkalmazás kódját, mely egy gombot jelenít meg és a gombot lenyomva a „Clicked!” szöveg jelenik meg egy Toast-ban!
- Hogy biztosítja az Android a lokalizáció támogatását?

# Tartalom

- UI építő elemek
  - > Layout (ViewGroup)
  - > View-k
- Animációk
- MotionLayout
- JetPack Compose
- Összefoglalás

# Android UI architektúra



# Layout-ok

# Android felhasználói felület felépítése

- minden elem a View-ból származik le
- Layout-ok (elrendezések):
  - > ViewGroup leszármazottak
  - > ViewGroup is a View-ból származik le!
- ViewGroup-ok egymásba ágyazhatók
- Saját View és ViewGroup is készíthető, illetve a meglevők is kiterjeszthetők

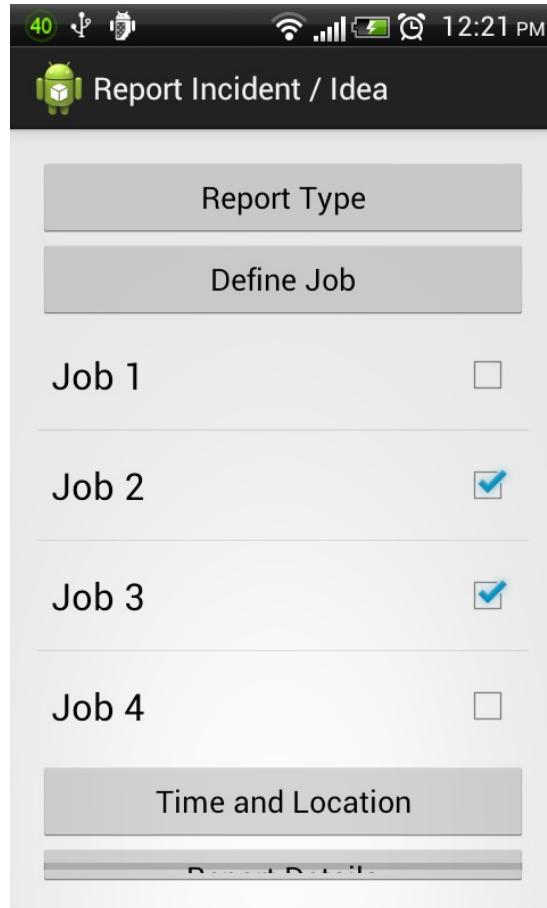
# Layout-ok (ViewGroup)

- LinearLayout
- RelativeLayout
- ConstraintLayout
- AbsoluteLayout (NEM használjuk!)
- GridLayout
- RecyclerView
- Teljes lista:
  - > <http://developer.android.com/reference/android/view/ViewGroup.html>

```
public class  
    RelativeLayout  
        extends ViewGroup  
  
java.lang.Object  
↳ android.view.View  
    ↳ android.view.ViewGroup  
        ↳ android.widget.RelativeLayout
```

# LinearLayout

- LinearLayout != Lista



# Súlyozás Layout tervezéskor

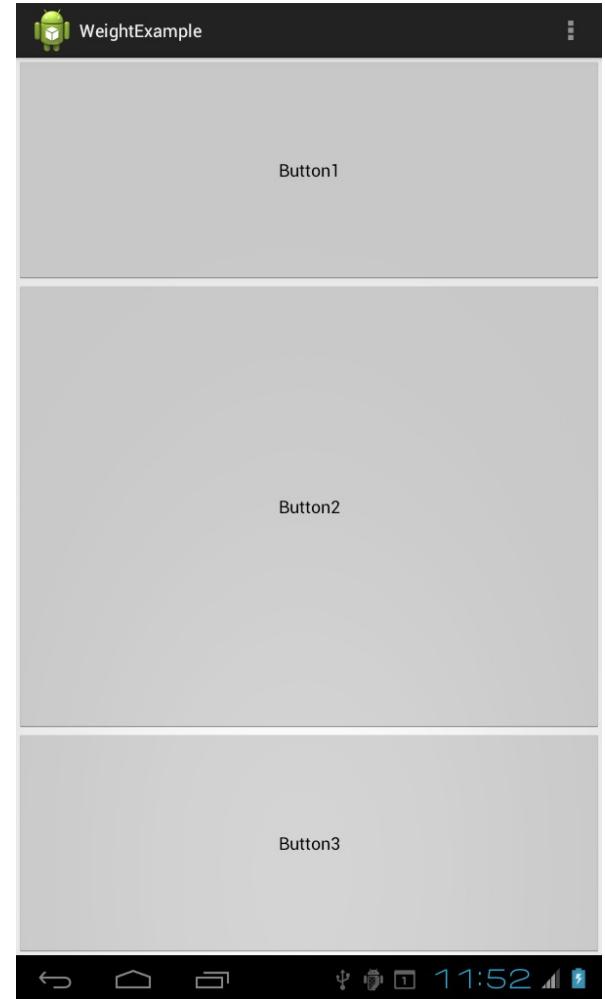
- Megadható egy layout teljes súly értéke (weightSum)
- Elemek súly értéke megadható és az alapján töltődik ki a layout
  - > layout\_weight érték
  - > A megfelelő width/height ilyenkor Odp legyen!
- Hasonló, mint HTML-ben a %-os méret megadás

# Layout súlyozás példa

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="4"
    android:orientation="vertical">

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button1" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:text="Button2" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button3" />

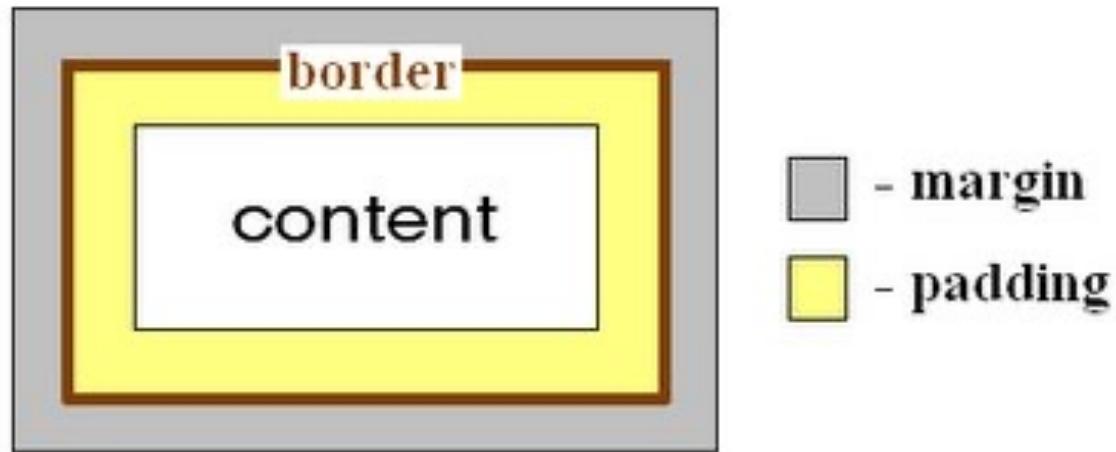
</LinearLayout>
```



# LinearLayout példák

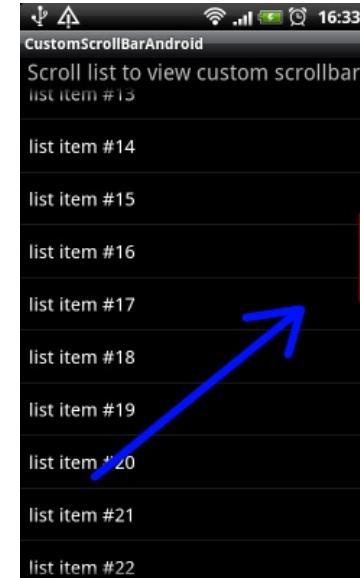
- Jellemző paraméterek:
  - > Margin, padding
  - > Gravity
  - > ScrollView
  - > Weight

# Padding és Margin



# ScrollView

- ScrollView és HorizontalScrollView
- Layout container, amely scrollozást tesz lehetővé, ha a benne levő tartalom „nagyobb”
- Nem kötelező a teljes képernyőt kitöltenie
- Egy layout/képernyő több ScrollView-t is tartalmazhat



# ScrollView példa

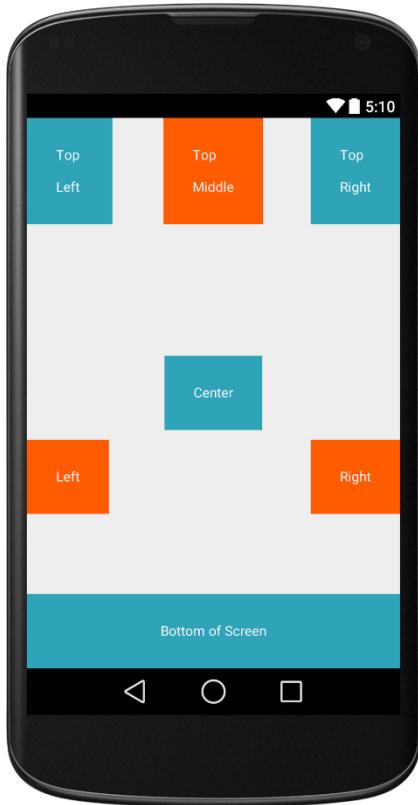
```
<ScrollView xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    android:fillViewport="false">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            android:src="@drawable/image" />

        ...
    </LinearLayout>
</ScrollView>
```

# RelativeLayout

- Elemek egymáshoz való viszonya definiálható
- Demo



# CoordinatorLayout, AppBarLayout

- CoordinatorLayout: továbbfejlesztett FrameLayout
- CoordinatorLayout fő feladatai:
  - > Felső szintű alkalmazás UI irányelv
  - > Konténer, mely támogatja a beépített elemek material stílushoz igazodó elhelyezkedését
- Behavior paraméterekkel meghatározható a kapcsolódó elemek elhelyezése
- AppBarLayout csatolható hozzá, mely a material design-hez illeszkedő scrollozást támogatja



# CONSTRAINTLAYOUT

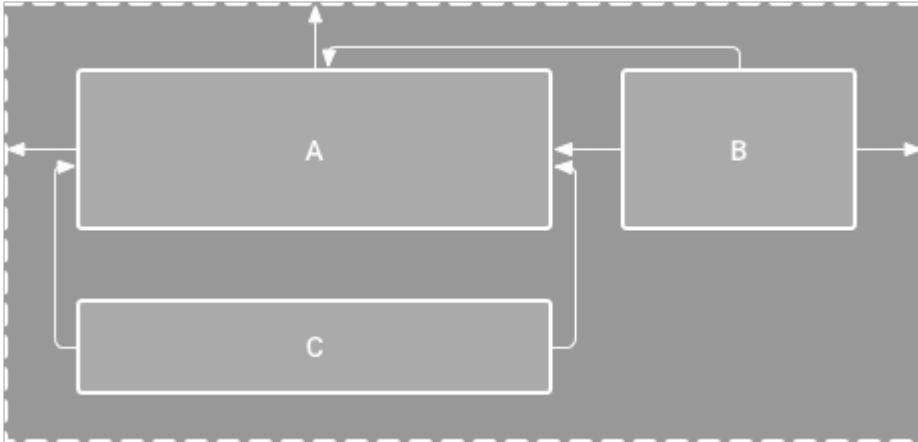
# Reszponzív felületek ConstraintLayout-al

- Összetett, komplex layout-ok flat view hierachiával
  - > Nincs szükség egymásba ágyazott layout-okra
- RelativeLayout-hoz hasonló
- Layout Editor támogatás
- Támogatás Android 2.3-tól (API Level 9)
- Komplex példák:
  - > <https://github.com/googlesamples/android-ConstraintLayoutExamples>

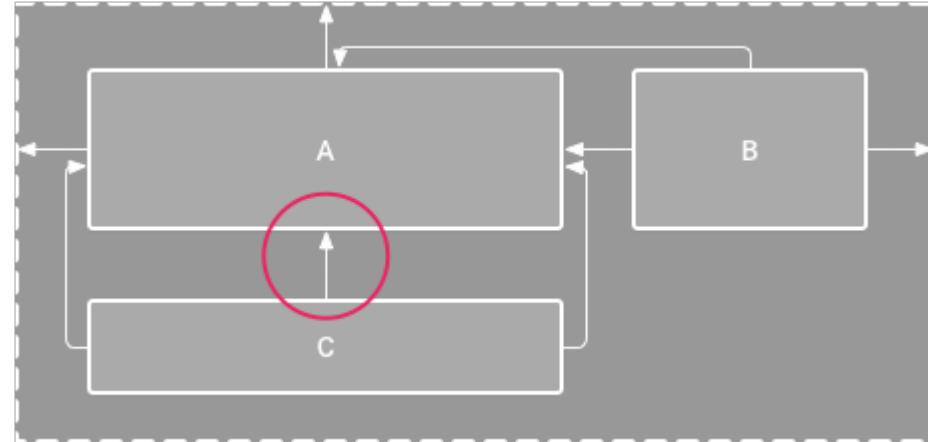
# Áttekintés

- Pozíció megadáshoz szükséges:
  - > Horizontális és vertikális „szabály” (constraint)
- minden szabály egy kapcsolat (connection)/igazítás (alignment):
  - > Egy másik view-hez képest
  - > Szülőhöz képest
  - > Egy láthatatlan sorvezetőhöz (guideline) képest
- Attól még, hogy a *LayoutEditor*-ban jól néz ki, nem biztos, hogy eszközön is jó lesz
- Android Studio jelzi a hiányzó szabályokat

Hibás:

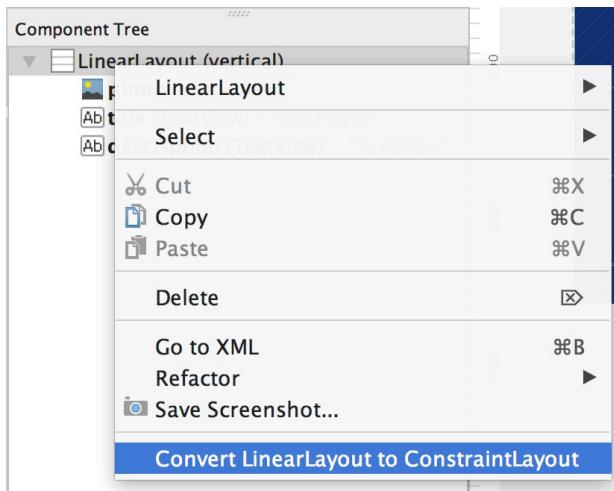


Helyes, mert C tudja, hogy A alatt van:



# ConstraintLayout eszközök

- Gradle import:
  - > compile 'com.android.support.constraint:constraint-layout:1.0.2'
- Automatikus átalakítás
  - > Nem tökéletes...



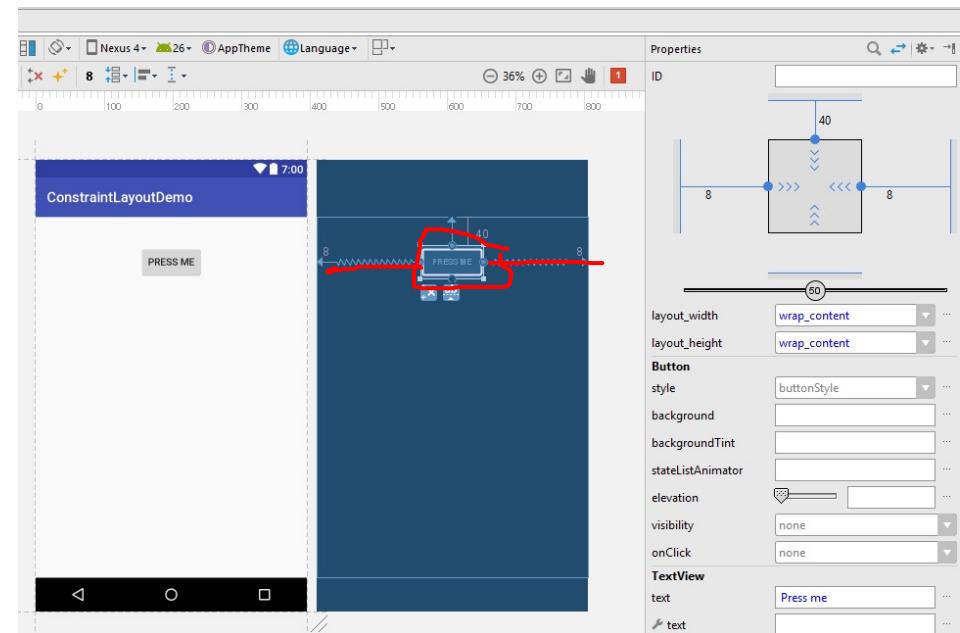
# ConstraintLayout használat

- Kötelező legalább egy horizontális és vertikális „szabály”

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    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"

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="8dp"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="40dp"
    />

</android.support.constraint.ConstraintLayout>
```



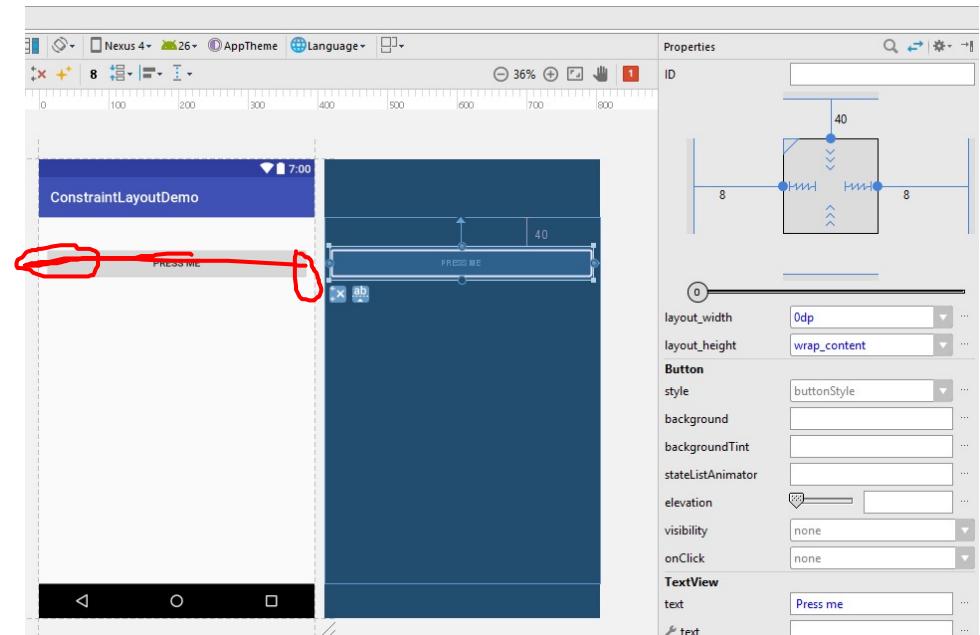
# Mekkora lesz a gomb mérete?

- Nem egyértelmű a szélesség, ellentétes szabályok, jele: 
  - Kettő közé helyezi
- Helyette automata méretezés:
  - `android:layout_width="0dp"`

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    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"

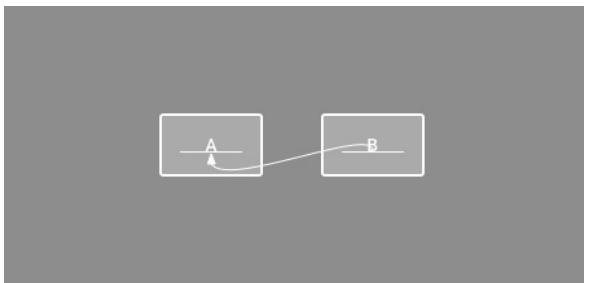
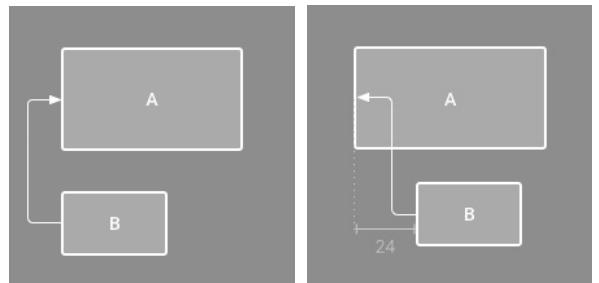
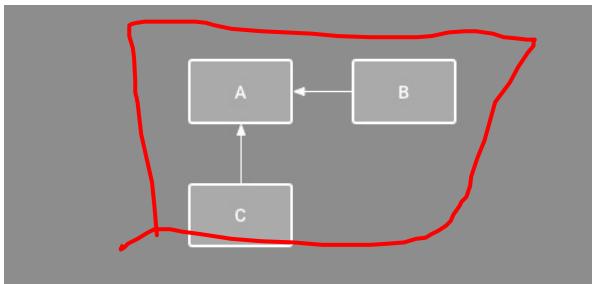
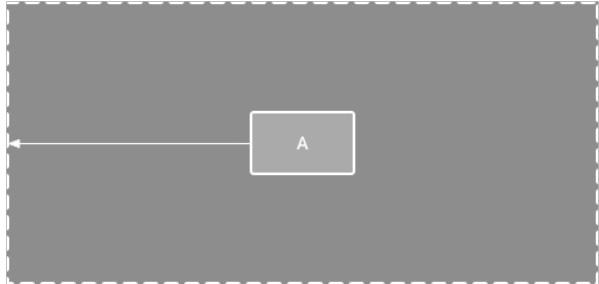
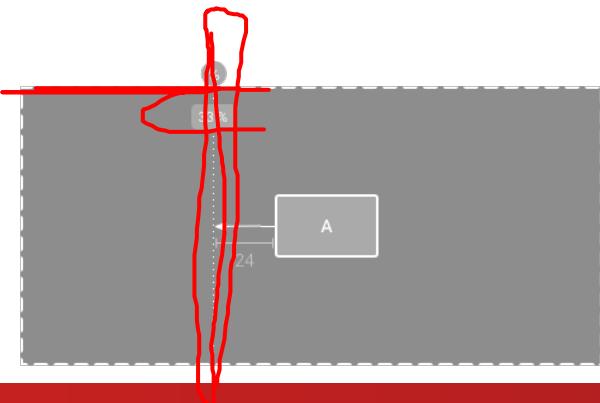
    <Button
        android:layout_width="0dp" // This line is circled in red
        android:layout_height="wrap_content"
        android:text="Press me"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="8dp",
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="40dp"
    />

</android.support.constraint.ConstraintLayout>
```



# Constraint lehetőségek

- Szülőhöz képest
- Másik View széleihez képest
- Másik View alapvonalához képest
- Guidelinehez (láthatatlan vezetővonalhoz)



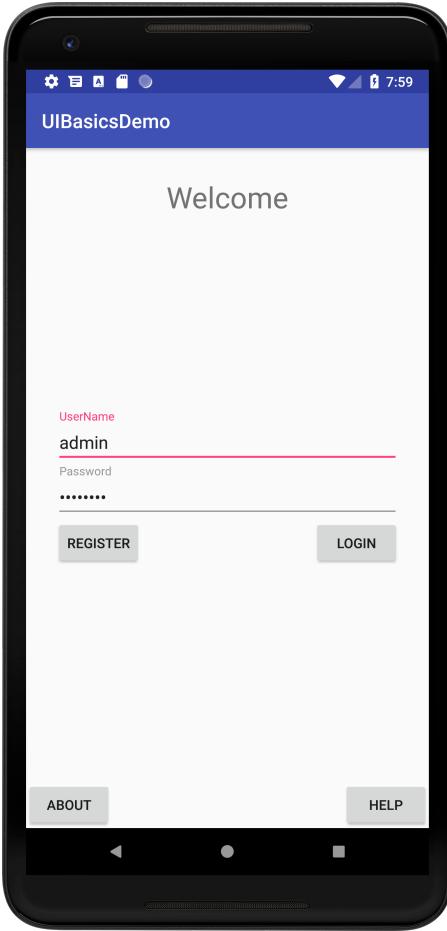
# ConstraintLayout teljesítmény

- <https://android-developers.googleblog.com/2017/08/understanding-performance-benefits-of.html>

# Gyakoroljunk



- Készítsünk egy Login képernyőt



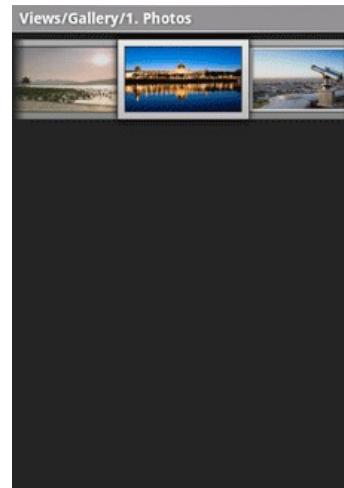
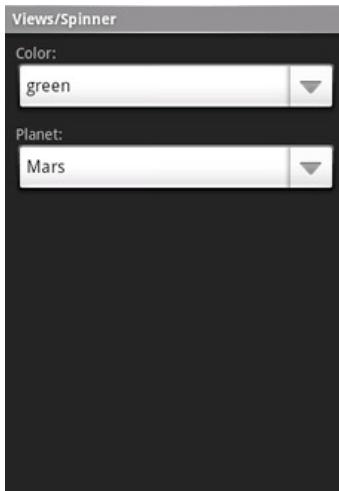
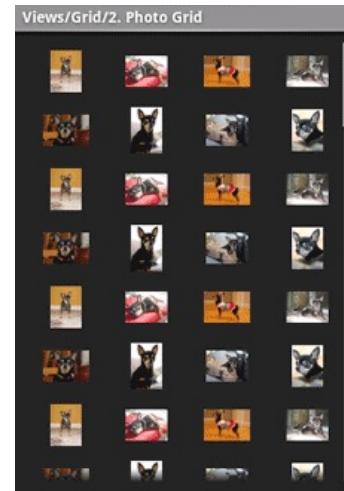
# Nézetek (Widgetek/"View"-k)

# View-k 1/2



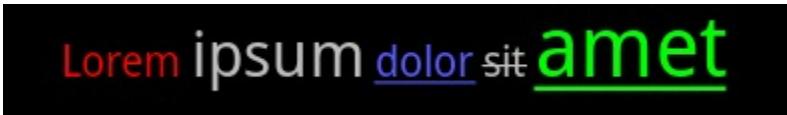
- *Button, EditText, CheckBox, RadioButton, ToggleButton*
- ImageButton*
- ListView*
- GridView*
- Spinner*
- AutoCompleteTextView*
- Gallery*
- ImageSwitcher*
- DatePicker, TimePicker*

# View-k 2/2



# API gazdagsága (globálisan igaz az Androidra)

- Hogy valósítanátok ezt meg? (nem sok *TextView* egymás után😊)



Lorem ipsum dolor sit amet

- Megoldás:
  - > <http://developer.android.com/reference/android/text/SpannableString.html>
  - > <http://androidcocktail.blogspot.hu/2014/03/android-spannablestring-example.html>

# Egyedi nézetek – külső könyvtárak

- <https://github.com/wasabeef/awesome-android-ui/>

# Dinamikus UI kezelés - LayoutInflater

- LayoutInflater feladata:
  - > XML-ben összeállított felületi elemek példányosítása
- Használati mód:

```
val myView = getLayoutInflater().inflate(  
    R.layout.activity_main, null)
```

# ANIMÁCIÓK

# Animációk

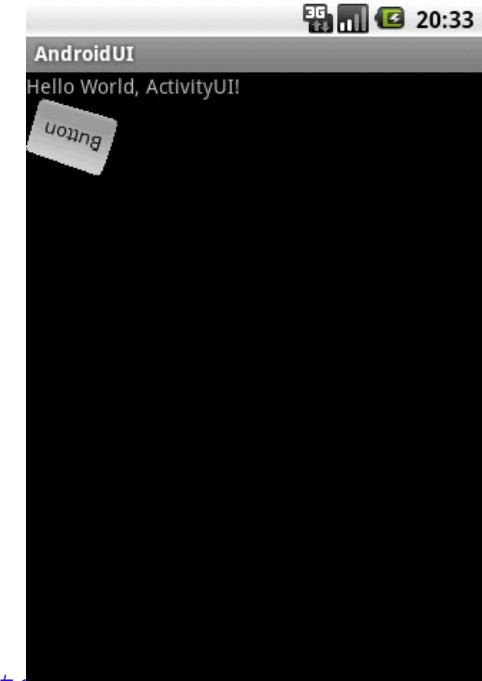
- Animációk támogatása
  - > XML erőforrás (*res/anim*)
  - > Programkód
- Layout animáció
  - > *Scale*
  - > *Rotate*
  - > *Translate*
  - > *Alpha*
- Hárrom fő típus:
  - > Tween animáció
  - > Frame animáció
  - > Property animator

# Tween animáció erőforrás

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator=
            "@android:anim/accelerate_interpolator"
        android:fromXScale="0.0"
        android:toXScale="1.0"
        android:fromYScale="0.0"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="1000" />

    <alpha android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="5000"/>

    <rotate
        android:interpolator="@android:anim/accelerate_interpolator"
        android:fromDegrees="0.0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="5000" />
</set>
```



# Tween animáció lejátszása

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        var demoAnim = AnimationUtils.loadAnimation(this,
            R.anim.demo_anim)

        btnAnim.setOnClickListener {
            btnAnim.startAnimation(demoAnim)
        }
    }
}
```

# MotionLayout

A new way to create animations on Android  
*Enyedi Péter*



**MOTION  
LAYOUT  
in  
Android**

# Animációs lehetőségek Androidon

## 1. ObjectAnimator

- > Reflection
- > minden animációhoz külön példány

```
ObjectAnimator.ofFloat(view, "translationX", 100f).apply
{
    duration = 2000
    start()
}
```

## 2. Animation osztály és leszármazottai

- > Egyetlen property-t animál, csak alap property-keket
- > Több példány is kell belőle egy AnimatorSet lejátszásához

# Animációs lehetőségek Androidon

## 3. ViewPropertyAnimator

- > Egyszerre több property animálása
- > Hatékonyabb, mint az ObjectAnimator

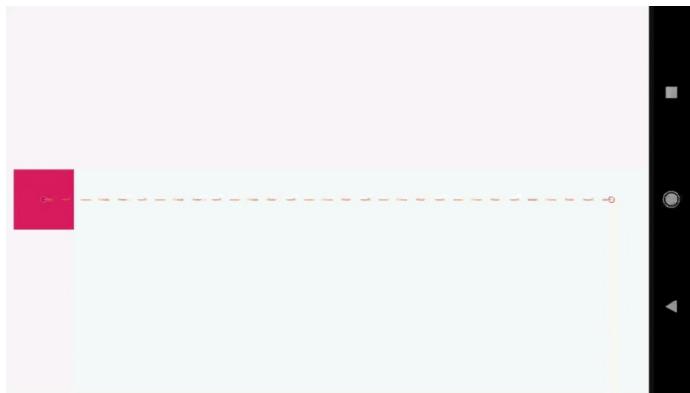
```
view.animate().x(50f).y(100f)
```

## 4. ValueAnimator

```
val va: ValueAnimator = ValueAnimator.ofFloat(0f, 3f)
val duration: Long = 3000
va.duration = duration
va.addUpdateListener { animation -> view.translationX = animation.animatedValue as Float
}
va.repeatCount = 5;
va.start();
```

# MotionLayout

- Google I/O 2018
- ConstraintLayout 2.0.1



MotionLayout

MotionScene

ConstraintSet

Transition

Listener (OnClick,  
OnSwipe)

KeyFrameSet

KeyPosition

KeyAttribute

CustomAttribute

KeyCycle

# MotionLayout - Felépítés

- Layout fájl

```
<androidx.constraintlayout.motion.widget.MotionLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    app:layoutDescription="@xml/demo_motion_scene"  
    app:motionDebug="SHOW_ALL">  
  
...  
</androidx.constraintlayout.motion.widget.MotionLayout>
```



# MotionLayout - Felépítés

- MotionScene
  - > ConstraintSet-ek
    - Többféleképpen d
    - Alap attribútumok
    - Constraint-ek

```
<ConstraintSet android:id="@+id/start">
    <Constraint
        android:id="@+id/imageView"
        android:layout_width="80dp"
        android:layout_height="80dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</ConstraintSet>

<ConstraintSet android:id="@+id/end">
    <Constraint
        android:id="@+id/imageView"
        android:layout_width="80dp"
        android:layout_height="80dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
</ConstraintSet>
```

# MotionLayout - Felépítés

- MotionScene
  - > Transition
    - Átmenet leírása
    - Listenerek
    - KeyFrameSet
      - KeyPosition
      - KeyAttribute
      - CustomAttribute
      - KeyCycle

```
<Transition
    app:constraintSetEnd="@+id/end"
    app:constraintSetStart="@+id/start"
    app:duration="3000">

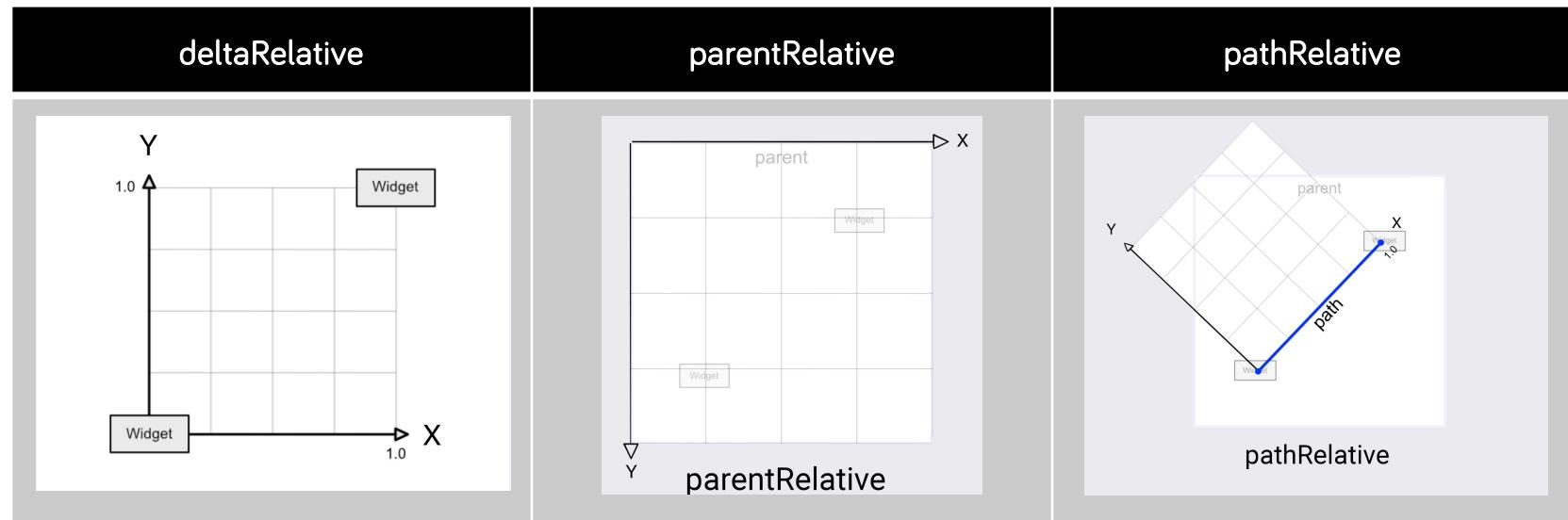
    <OnClick
        app:clickAction="toggle"
        app:targetId="@+id/imageView" />

    <KeyFrameSet>
        <KeyPosition
            app:framePosition="50"
            app:keyPositionType="parentRelative"
            app:motionTarget="@+id/imageView"
            app:percentY="0.5" />
    </KeyFrameSet>

</Transition>
```

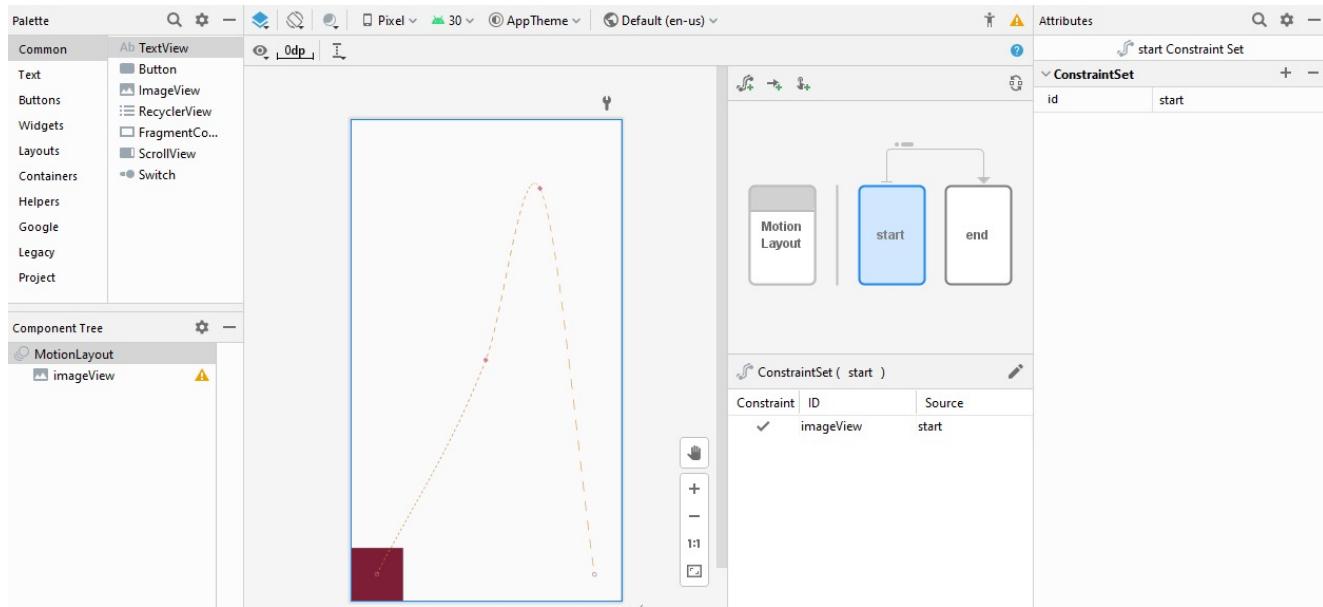
# MotionLayout - Felépítés

- KeyPosition



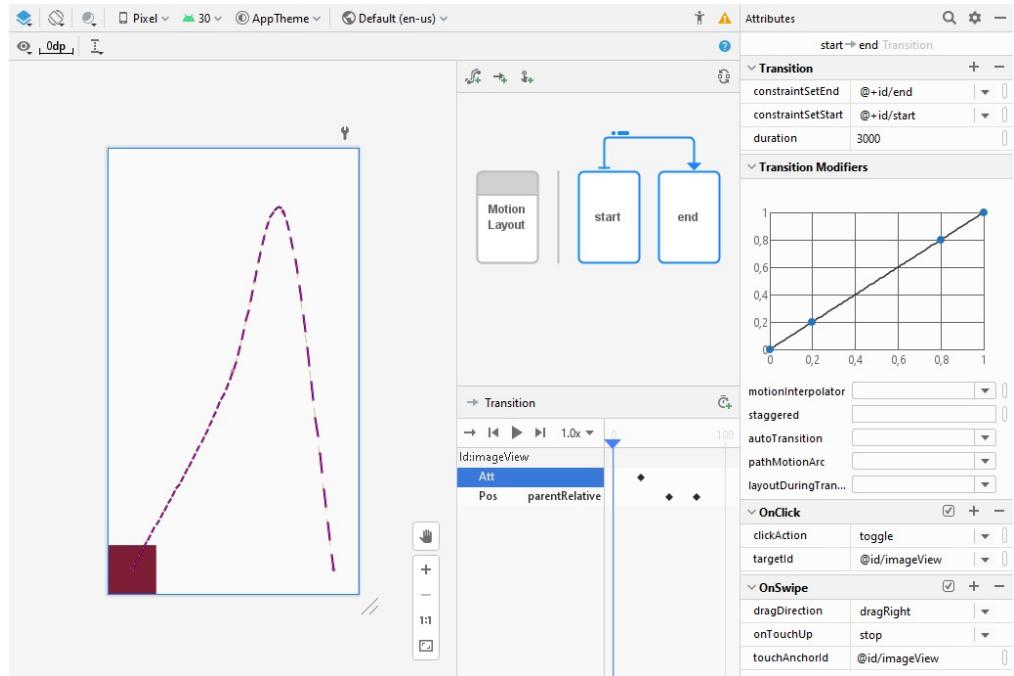
# Motion Editor

- Android Studio 4.0 Canary 1-től



# Motion Editor

- Transition editor



# MotionLayout hasznos linkek

- <https://developer.android.com/training/constraint-layout/motionlayout>
- <https://developer.android.com/training/constraint-layout/motionlayout/examples>
- <https://blog.autsoft.hu/motionlayout-a-new-way-to-create-animations-on-android/>

# Transition animációk

- Érdekesség:
  - > <https://www.uplabs.com/posts/shared-element-transition-kotlin>

# JETPACK COMPOSE

# Jetpack compose

- Modern toolkit UI készítéshez
  - > Jelenleg beta verzió
    - (de telefonon fut már)
  - > A fejlesztő Kotlin kóddal leírja a UI paramétereit és a Compose motor generálja a felületet
  - > Könnyű a UI frissítése az alkalmazás állapota alapján
  - > Erőforrásokat (pl képek) ugyanúgy használhat
- Csomag része
  - > Eszközök
  - > Kotlin API-kat
- Előnyök
  - > Kevesebb kód
  - > Nincs szükség XML layout-ra
  - > Nincs szükség UI widgetek készítésére
  - > A UI elemek kódból készíthetők
  - > Könnyebb az újrafelhasználhatóság
  - > Kompatibilis a meglévő UI toolkit-el (XML – layout megoldással)

# Composable függvények

- Segítségükkel készíthetők UI elemek Kotlin kódból
  - > Alakzat és adat függőség megadása
- @Composeble annotáció a függvények elején
- Egymásba ágyazható UI elemek

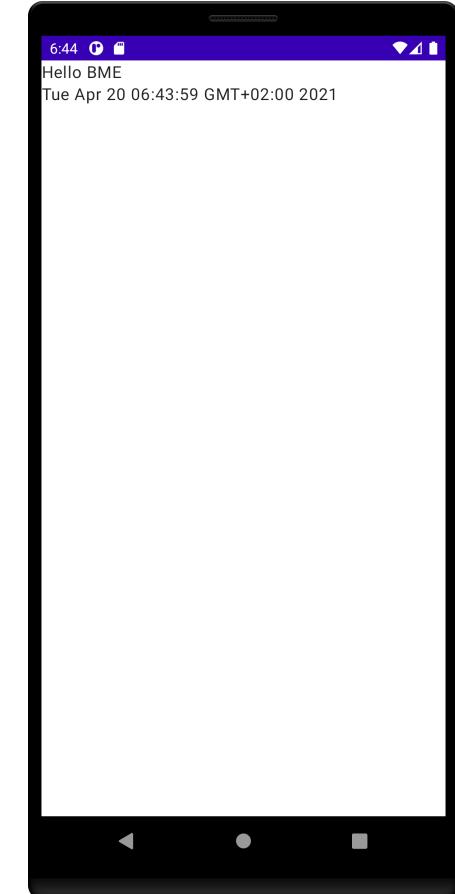
# Jetpack Compose – HelloWorld

```
class DemoActivity : ComponentActivity(){  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            MaterialTheme {  
                HelloWorld()  
            }  
        }  
    }  
  
    @Composable  
    fun HelloWorld() {  
        Column {  
            Text("Hello BME")  
            Text(Date(System.currentTimeMillis()).toString())  
        }  
    }  
}
```

*ComponentActivity  
leszármazott*

*setContentView()  
helyett setContent*

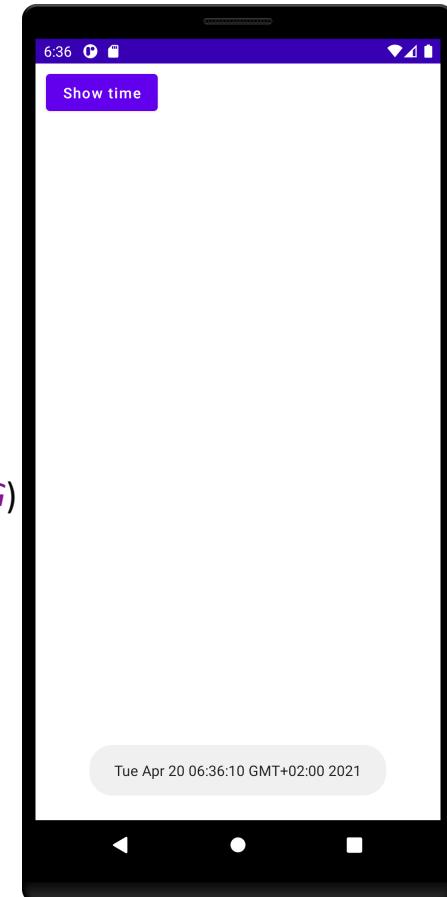
*@Composeable  
függvény*



# Jetpack Compose – Események

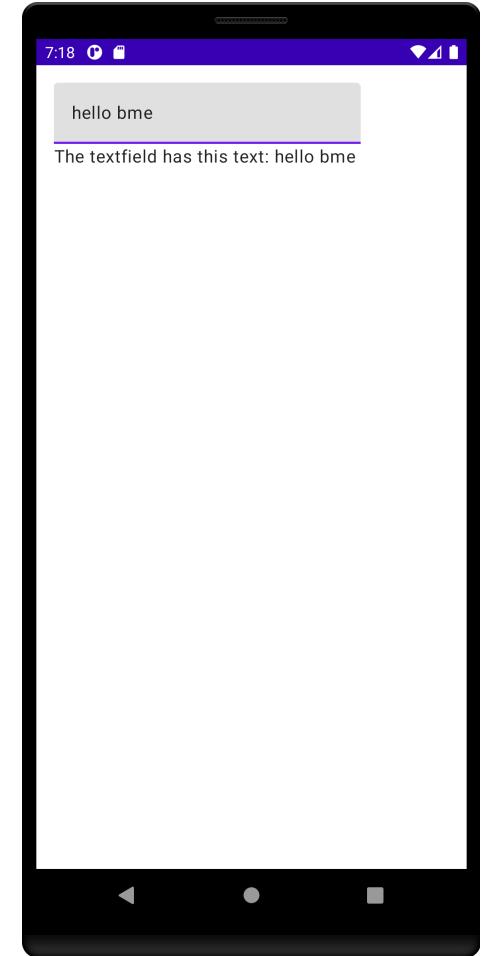
```
class DemoActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ButtonShowTime()
        }
    }

    @Composable
    fun ButtonShowTime() {
        Button(
            onClick = {
                Toast.makeText(this, Date(System.currentTimeMillis()).toString(), Toast.LENGTH_LONG)
                    .show()
            },
            modifier = Modifier.padding(Dp(10f))
        ) {
            Text("Show time")
        }
    }
}
```



# Jetpack Compose – TextField (input)

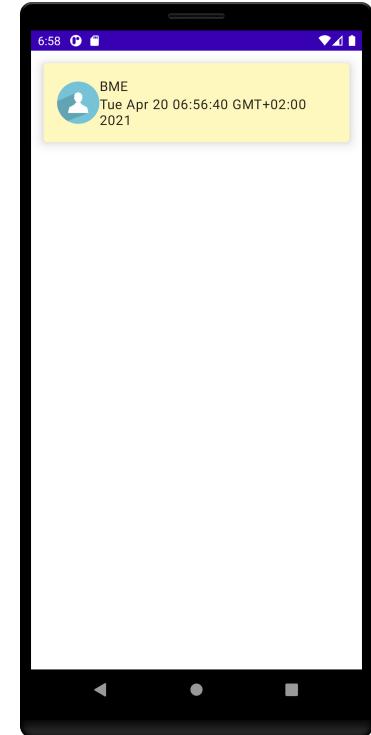
```
class DemoActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            MaterialTheme {  
                TextFieldDemo()  
            }  
        }  
    }  
  
    @Composable  
    fun TextFieldDemo() {  
        Column(Modifier.padding(16.dp)) {  
            val textState = remember { mutableStateOf(TextFieldValue()) }  
            TextField(  
                value = textState.value,  
                onValueChange = { textState.value = it }  
            )  
            Text("The textfield has this text: " + textState.value.text)  
        }  
    }  
}
```



# Jetpack Compose – Képek és Card

```
@Composable
fun DemoCard(name: String) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(15.dp)
            .clickable {
                Toast.makeText(
                    this,
                    Date(System.currentTimeMillis()).toString(),
                    Toast.LENGTH_LONG
                ).show()
            },
        elevation = 10.dp,
        backgroundColor = Color(255,248,190)
    ) {
        Row(verticalAlignment = Alignment.CenterVertically, modifier =
        Modifier.padding(16.dp)) {
            Image(
                painter = painterResource(R.drawable.person),
                contentDescription = "Person",
                Modifier.size(50.dp, 50.dp),
                contentScale = ContentScale.Fit
            )
            Column {
                Text(name)
                Text(Date(System.currentTimeMillis()).toString())
            }
        }
    }
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
        MaterialTheme {
            DemoCard(name = "BME")
        }
    }
}
```



# További anyagok

- <https://developer.android.com/jetpack/compose/tutorial>
- <https://developer.android.com/jetpack/compose/state>
- <https://foso.github.io/Jetpack-Compose-Playground/>
- <https://joebirch.co/android/exploring-jetpack-compose-card/>

# Összefoglalás

- UI építő elemek
  - > Layout (ViewGroup)
  - > View-k
- Animációk
- MotionLayout
- JetPack Compose
- Összefoglalás

# Kérdések

