DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Entwicklung mobiler Applikationen

WS2022/2023

Jan Brodhaecker | Oktober 2022

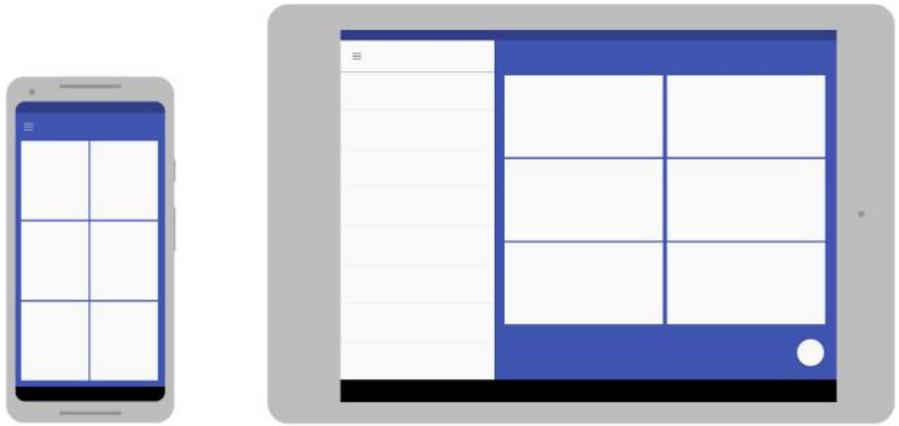
Agenda

- Activities / Layouts
  - DataBinding
  - ViewBinding
  - Fragments
  - Jetpack Compose

Android Basics

Activities - Layouts

https://wish.link/31xgrKY



https://developer.android.com/training/multiscreen/screensizes

https://windowsunited.de/exklusivbilder-samsung-galaxy-fold-2-ultimatives-falt-flaggschiff-erwacht-zum-leben/

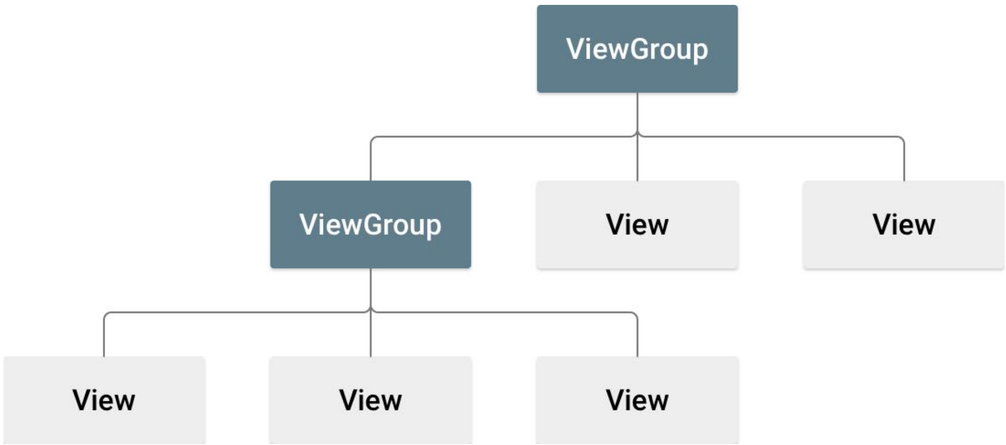
https://www.androidauthority.com/best-small-android-phones-782746/

Android Basics

Activities - Layouts

```
<TextView
 android:id="@+id/text_view_hello_world"
...
```

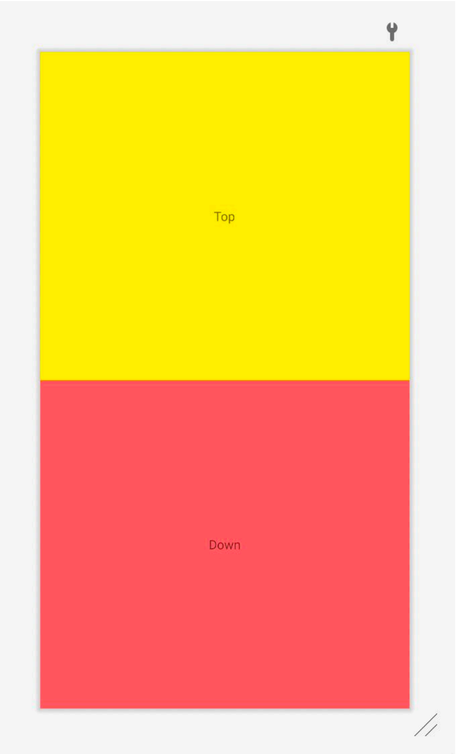
- Layout definiert die Struktur eines User-Interfaces (bspw. einer Activity)
- Hierachie von Views und ViewGroups
  - View - Inhalt sichtbar für den Benutzer
  - ViewGroup - Container
- Layouts werden in xml definiert, bzw. können zur Laufzeit instanziiert werden
- Elemente in einem Layout sollten eine id haben



https://developer.android.com/guide/topics/ui/declaring-layout

Android Basics

Activities - Layouts



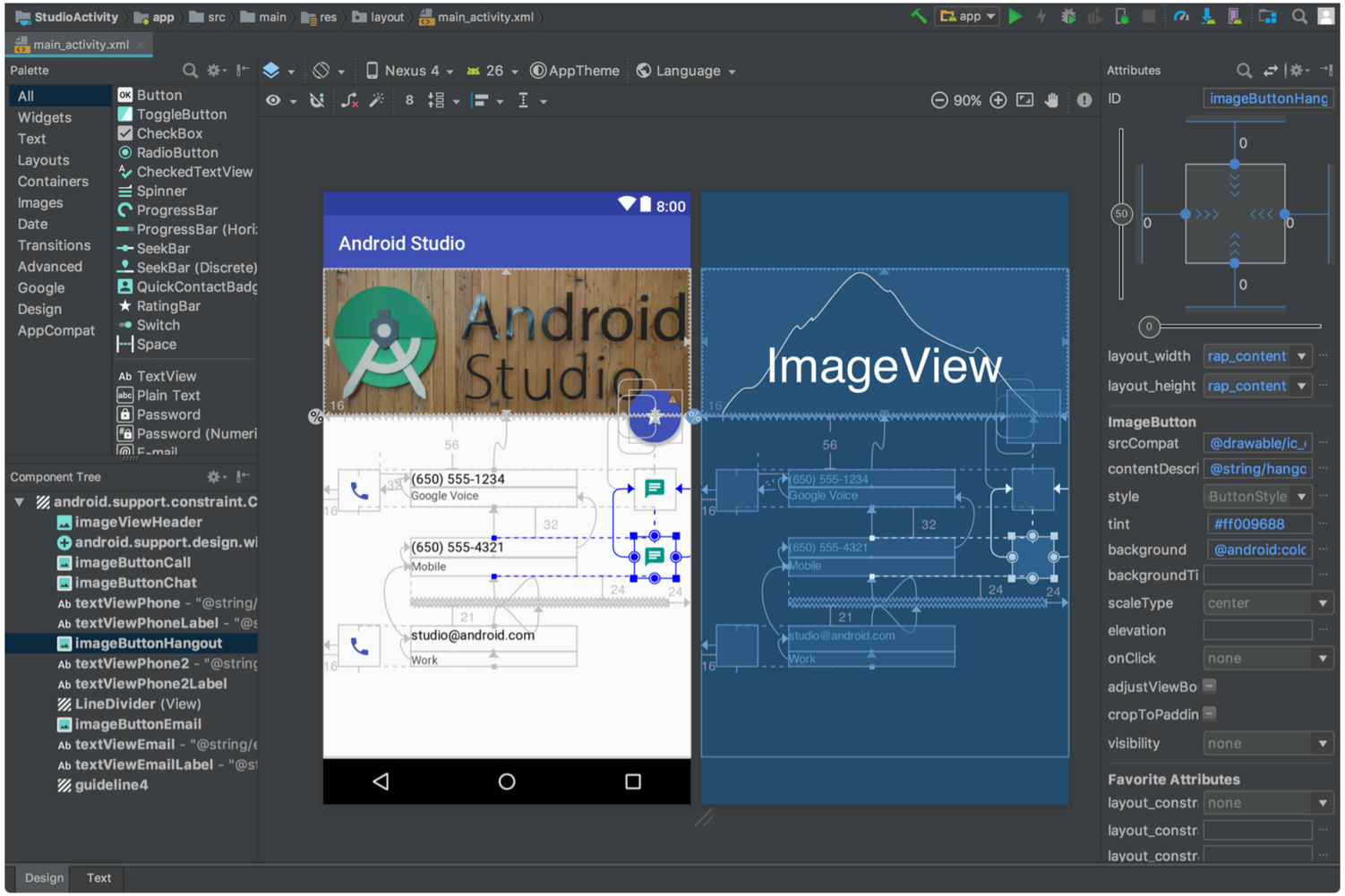

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"
            android:background="#fff000">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Top"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"
            android:background="#ff6666">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Down"/>
        </LinearLayout>
    </LinearLayout>
</layout>
```

Android Basics

Activities - Layouts

https://developer.android.com/training/multiscreen/screensizes

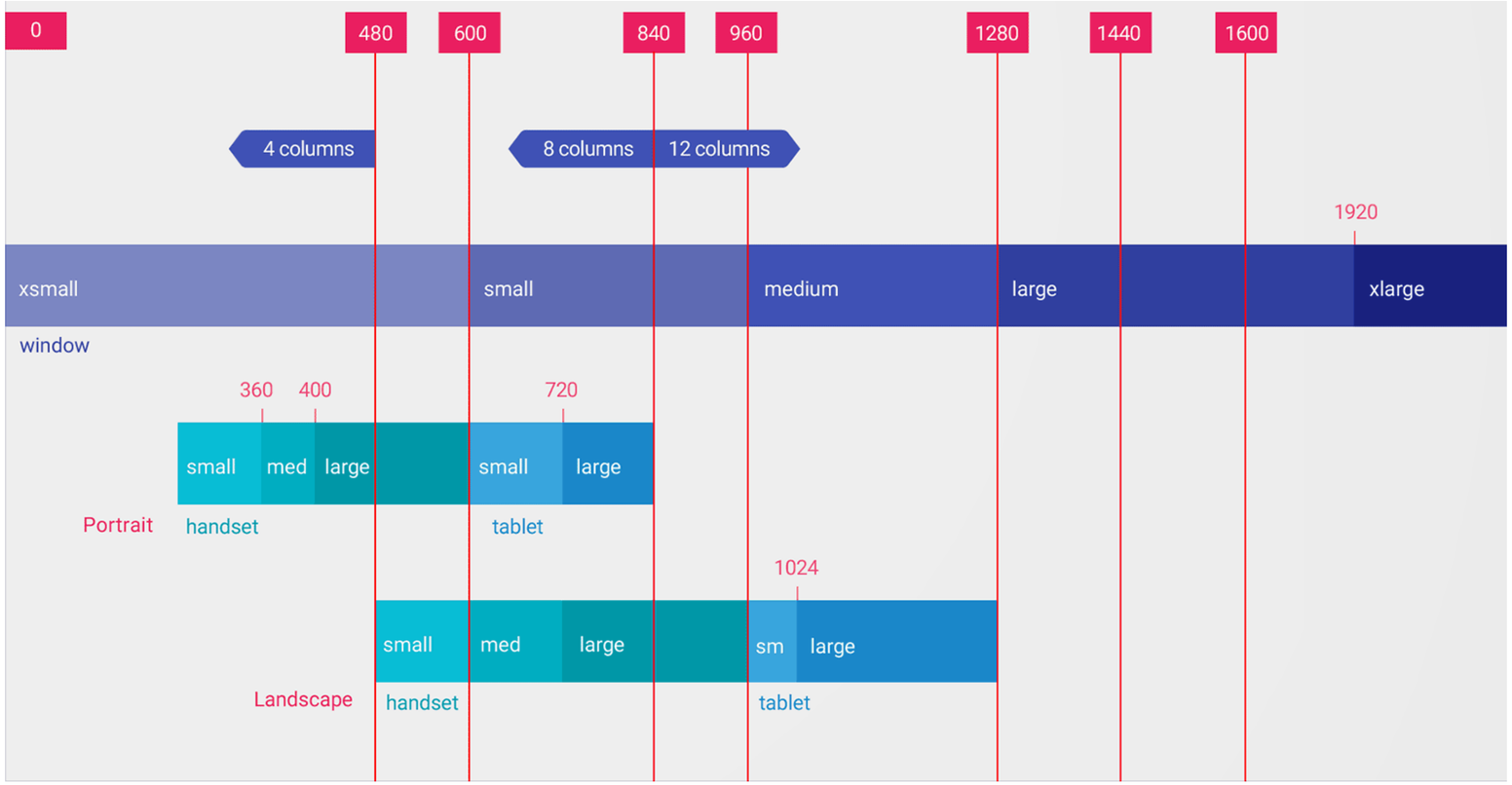
- Constraint Layouts
  - responsiveness
  - Elemente werden relativ zum jeweiligen Parent angeordnet
  - volle Integration in Android Studio



Android Basics

Activities - Layouts

- Android lädt Layouts anhand bestimmter Qualifier



https://developer.android.com/training/multiscreen/screensizes

Android Basics

Activities - Layouts

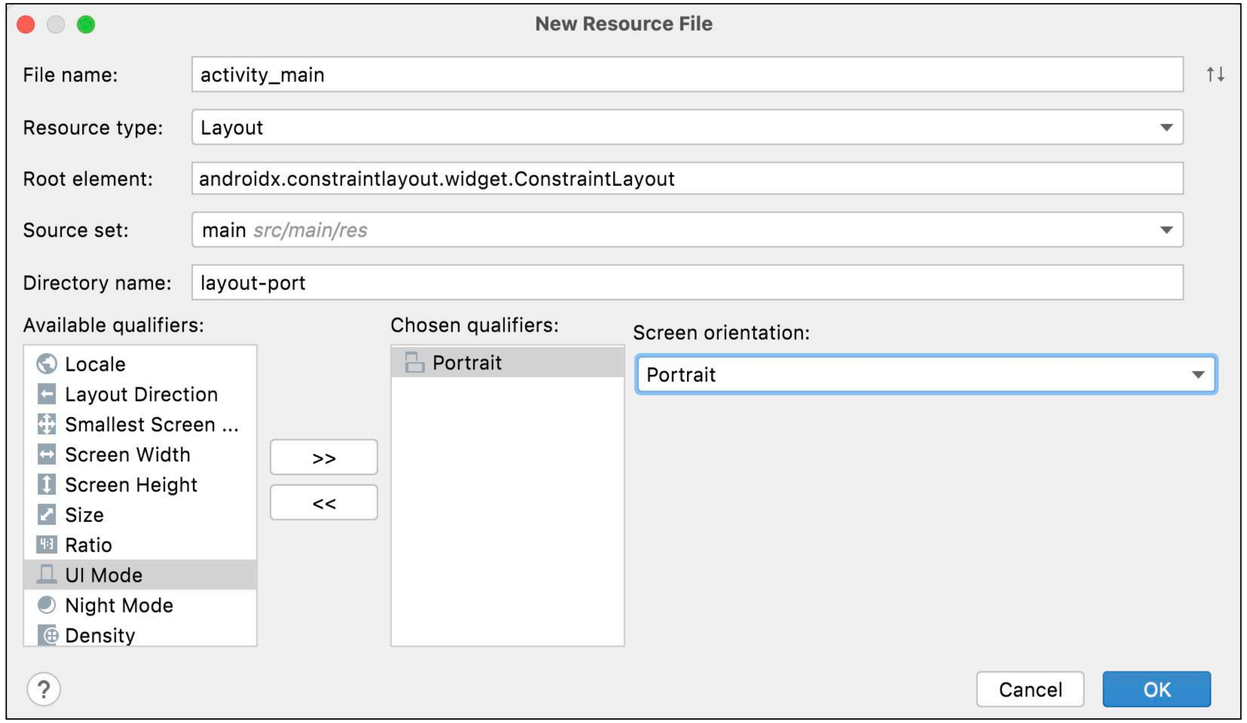
```
res/layout/main_activity.xml                # For handsets
res/layout-land/main_activity.xml           # For handsets in landscape
res/layout-sw600dp/main_activity.xml        # For 7" tablets
res/layout-sw600dp-land/main_activity.xml   # For 7" tablets in landscape
```

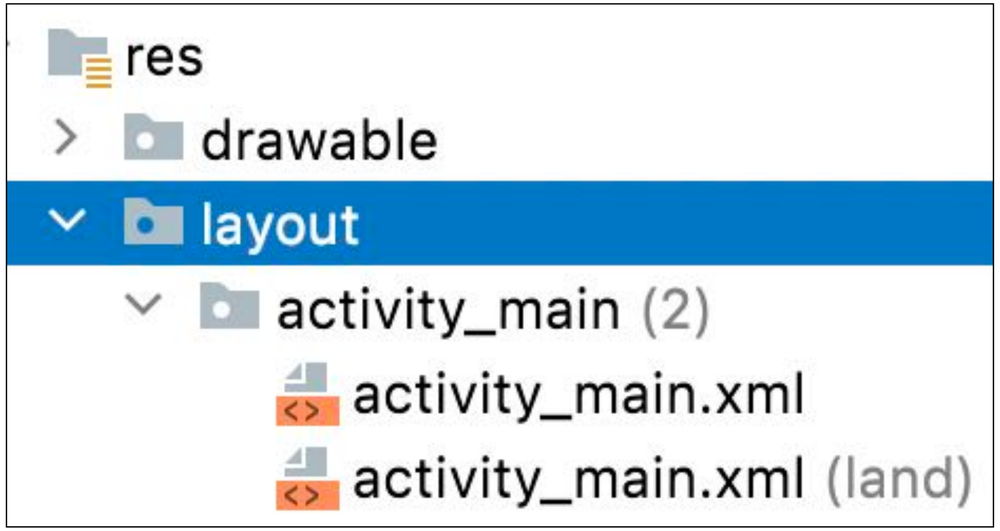
```
res/layout/main_activity.xml          # For handsets (smaller than 640dp x 480dp)
res/layout-large/main_activity.xml    # For small tablets (640dp x 480dp and bigger)
res/layout-xlarge/main_activity.xml   # For large tablets (960dp x 720dp and bigger)
```

https://developer.android.com/training/multiscreen/screensizes

Android Basics

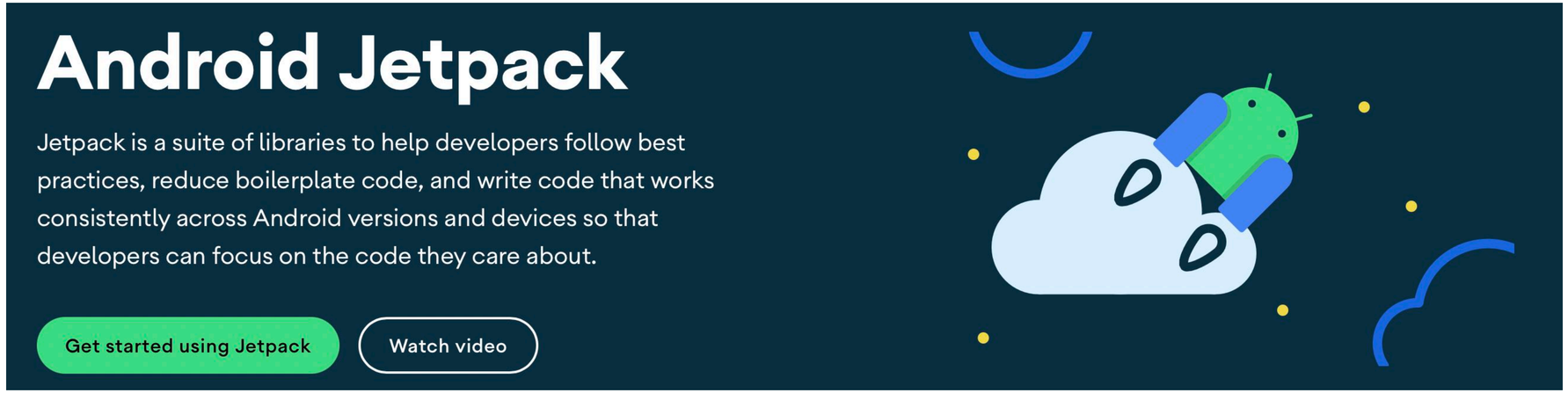
Activities - Layouts





Android Basics

Exkurs: Jetpack



`.androidx.`
im Packet beschreibt Jetpack Bibliothek

https://developer.android.com/jetpack

https://developer.android.com/jetpack/androidx/explorer

Android Basics

Activities - Layouts / View Binding

- generiert automatische Referenzen auf Layout-Elemente
- bietet u.a folgende Vorteile
  - Null-Safe: keine Initialisierung nötig
  - Type-Safe: korrekte Typisierung anhand der XML-Datei

```
android {
  ...
  buildFeatures {
    viewBinding = true
  }
}
```

Part of Android Jetpack.

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.textViewCounter.text = "Hello World!"
    }

...
```

https://developer.android.com/topic/libraries/view-binding

Android Basics

Activities - Layouts / Data Binding (1)

Part of Android Jetpack.

- bindet UI-Komponenten (Labels, EditText, …) in Layouts an Daten-Modelle (POJOs)
- vermeidet Fehler (bspw. auto-completion)
- vermeidet Komplexität in Activities/Fragments
- erlaubt Binding **Expressions** & **Two-Way-Data-Binding**

```
android {
  ...
  buildFeatures {
    dataBinding = true
  }
}
```

https://developer.android.com/topic/libraries/data-binding

Android Basics

Activities - Layouts / Data Binding (2)

Part of Android Jetpack.

```
data class Model(val text: String) {
...
```

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        binding.model = Model("Hello World!")
        setContentView(binding.root)

...
```

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

    <data>
        <variable name="model" type="de.dhbw.Model" />
    </data>

    <TextView
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:text="@{model.text}"/>
</layout>
```

https://developer.android.com/topic/libraries/data-binding

Android Basics

Activities - Layouts / Data Binding (3)

Part of Android Jetpack.

- Binding Expressions
  - mathematisch + - / * %
  - String konkatinierung +
  - logisch && ||
  - Vergleiche == > < >= <=
  - …

```
android:text="@{String.valueOf(index + 1)}"
android:visibility="@{age > 13 ? View.GONE : View.VISIBLE}"
android:transitionName='@{"image_" + id}'
```

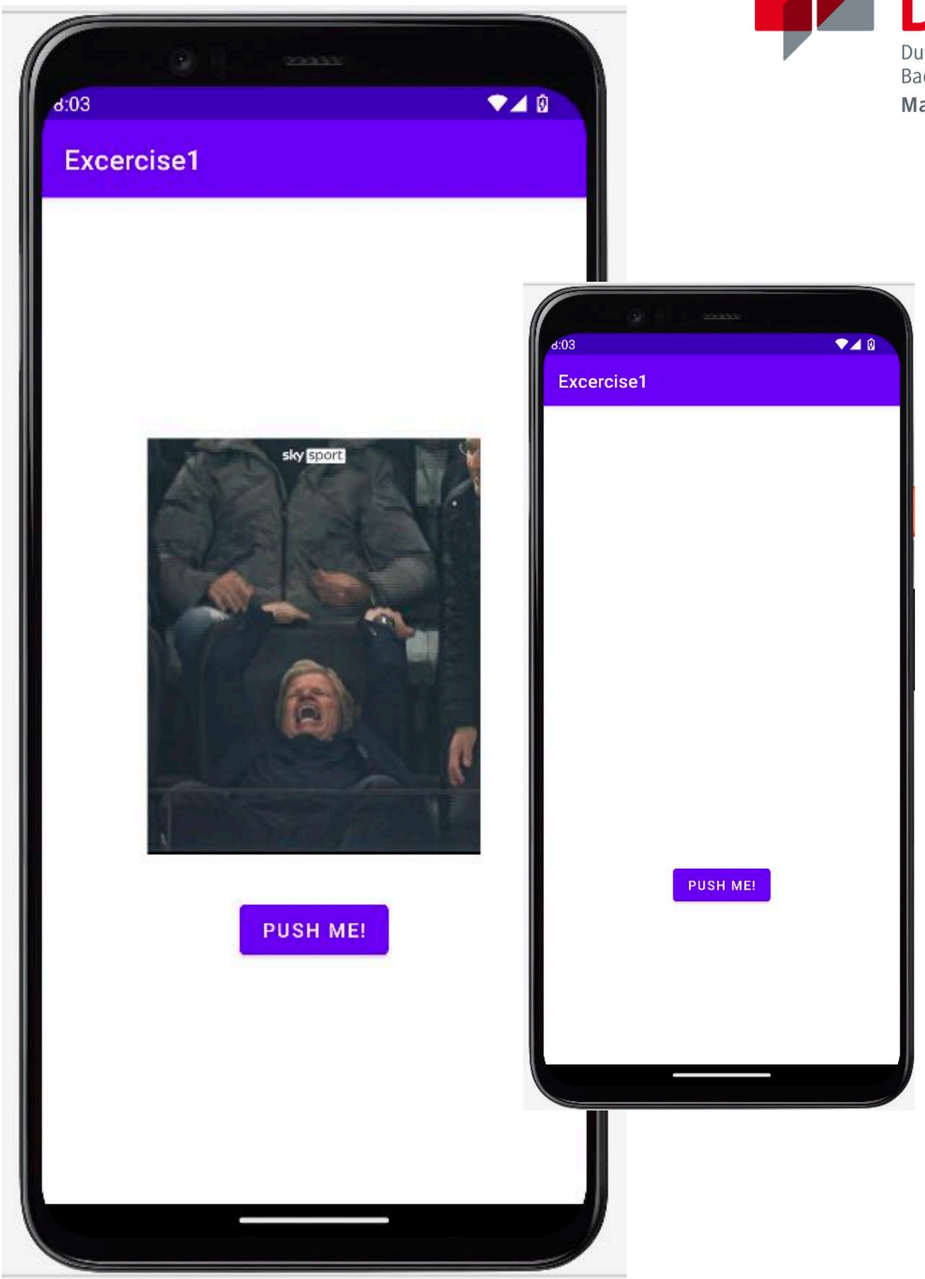
https://developer.android.com/topic/libraries/data-binding/expressions

https://developer.android.com/topic/libraries/data-binding

Android Basics

Übung 1 - Data Binding

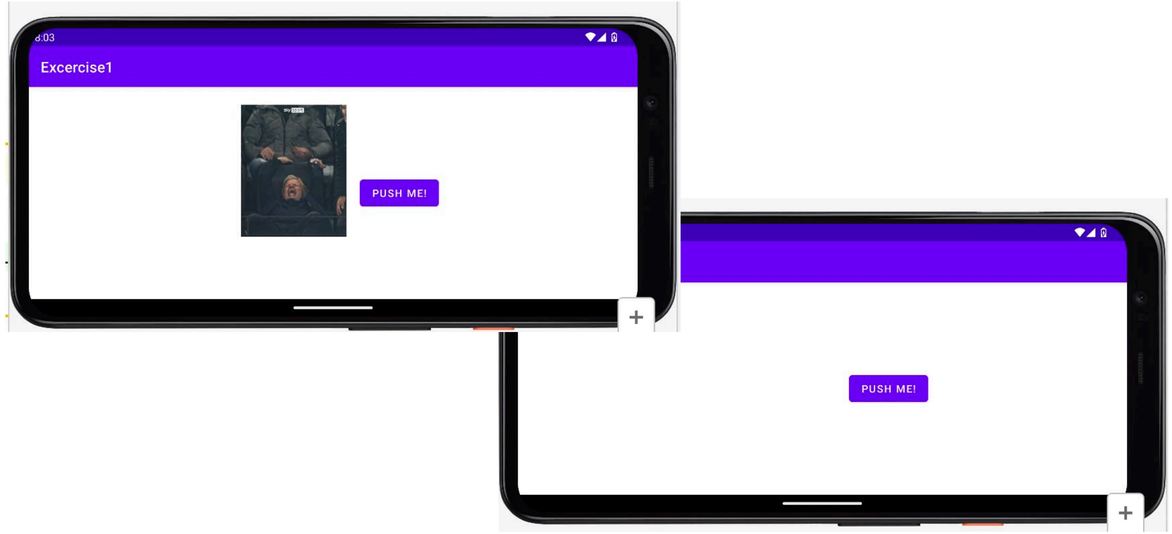
- Realisieren Sie das nebenstehende Projekt mit Hilfe von DataBinding
  - `ViewModel` anlegen
  - `Button` und `ImageView` hinzufügen
  - Sobald der Button geklickt wird, soll das Meme erscheinen



Android Basics

Übung 1.1 - Layouts

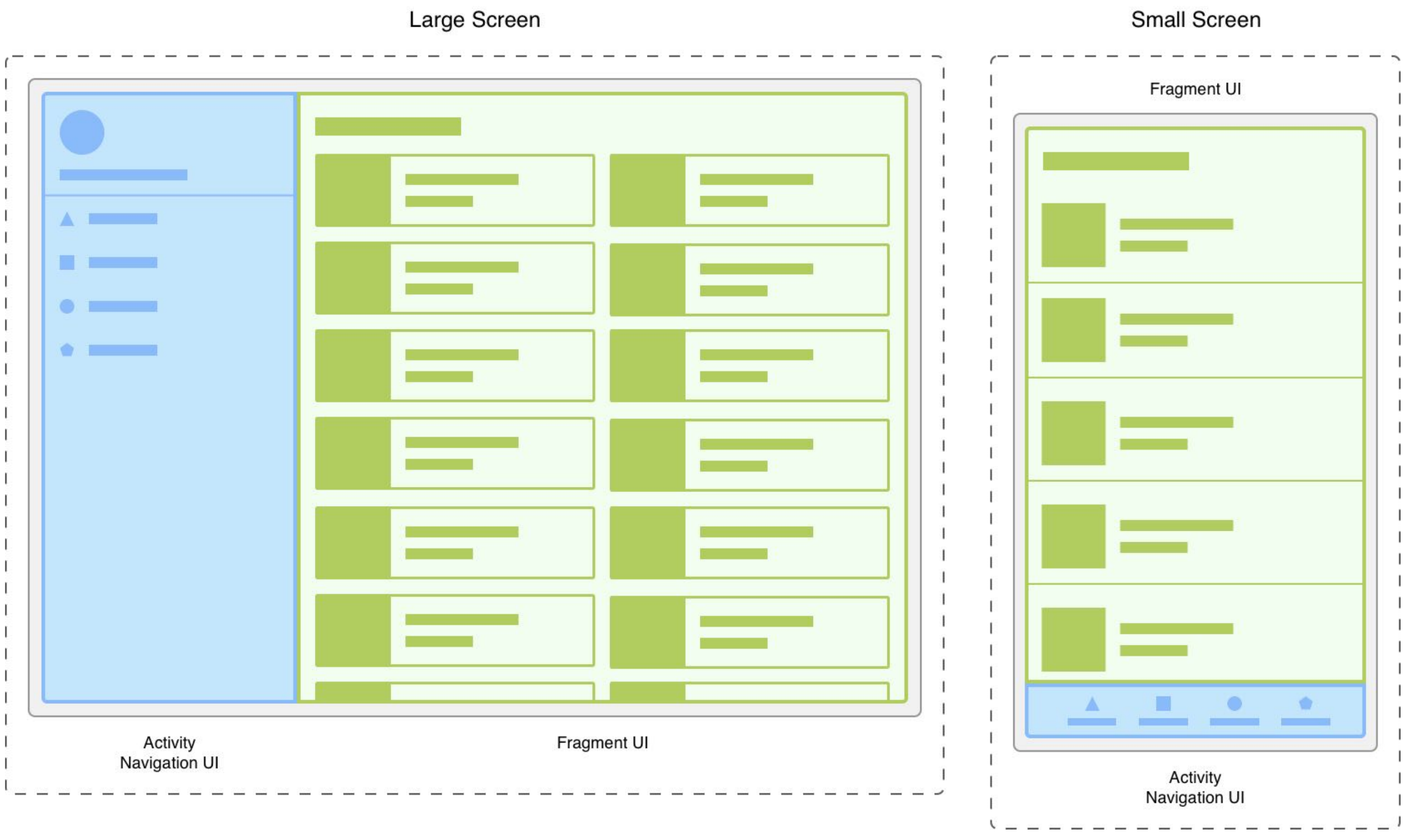
- wechseln Sie das Layout, sobald das Gerät die Orientierung ändert:
  - der Button soll links neben dem Meme stehen



Android Basics

Activities - Fragments

- Fragments sind wieder-verwendbare Teile eines UIs
- definiert und verwaltet eigenes Layout
- eigener Lifecycle
- müssen von Fragment oder einer Activity gehostet werden



https://developer.android.com/guide/fragments

Android Basics

Activities - Fragments

ein Fragment kann direkt geladen werden …

```
<androidx.fragment.app.FragmentContainerView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="de.dhbw.HelloWorldFragment"/>
```

… oder programmatisch …

```
class ExampleActivity : AppCompatActivity(R.layout.example_activity) {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    if (savedInstanceState == null) {
      val bundle = bundleOf("some_int" to 0)
      supportFragmentManager.commit {
        setReorderingAllowed(true)
        add<ExampleFragment>(R.id.fragment_container_view, args = bundle)
      }
    }
  }
}
```

https://developer.android.com/guide/fragments

Android Basics

Layouts: Jetpack Compose

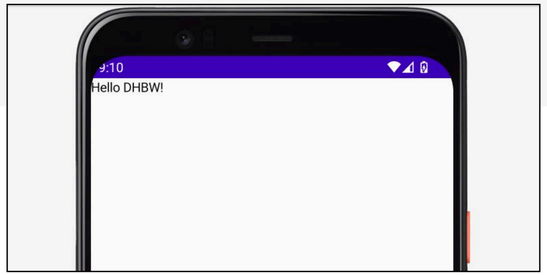
- modernes Toolkit um moderne UIs zu bauen
- soll Entwicklung von UIs beschleunigen und vereinfachen
- … soll XML und Layout-Editor überflüssig machen
- UIs sollen mit Hilfe von `composable` Funktionen entwickelt werden
- keine Trennung mehr zwischen UI und Business-Logik
- https://developer.android.com/jetpack/compose

Android Basics

Layouts: Jetpack Compose

- `setContent{ … }` - definiert Layout
- `Text()` composable Funktion
- Kotlin Compiler übersetzt composable Funktionen in UI

```kotlin
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Text("Hello DHBW!")
        }
    }
}
```

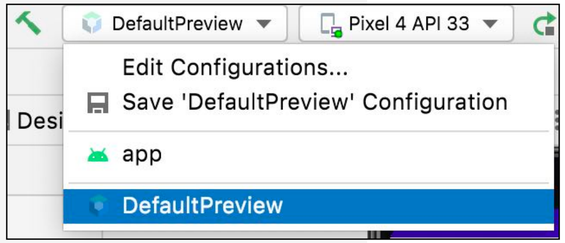


Android Basics

Layouts: Jetpack Compose

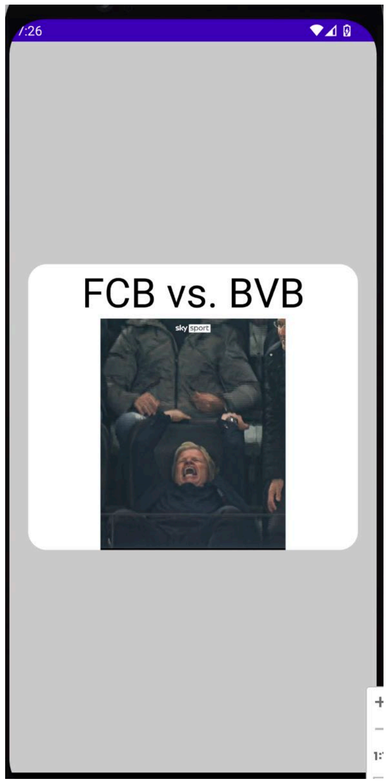
- `@Composable` markiert Funktionen als composable
- nur composable Funktionen können composable Funktionen rufen
- `@Preview` erlaubt statische Ansicht des UIs

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MessageCard(name = "Runtime")
        }
    }

    @Composable
    fun MessageCard(name: String) {
       return Text("Hello $name")
    }

    @Preview
    @Composable
    fun DefaultPreview() {
       return MessageCard(name = "Preview")
    }
}
```



Android Basics

Layouts: Jetpack Compose

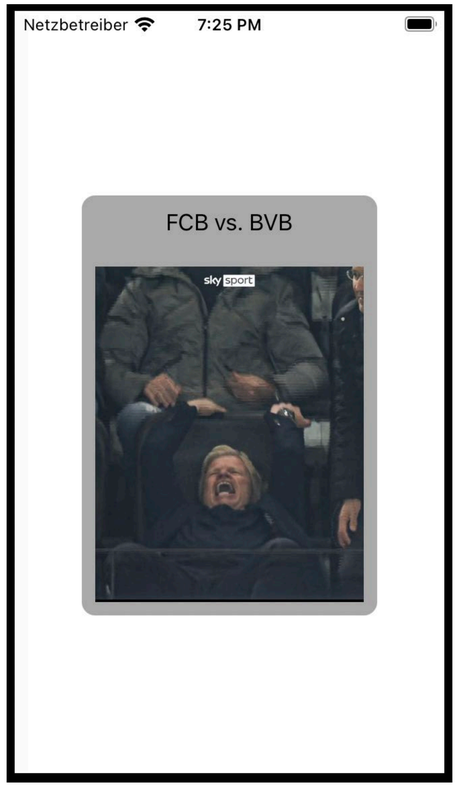



```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Box(modifier = Modifier.fillMaxSize().background(Color.LightGray),
                contentAlignment = Alignment.Center) {

              MemeCard(model = MemeCardModel("...", "FCB vs. BVB"))
            }
        }
    }

    @Composable
    fun MemeCard(model: MemeCardModel) {
        Column(modifier = Modifier.fillMaxWidth()) {
            Column(modifier =
                Modifier.fillMaxWidth().padding(20.dp).background(Color.White)) {
                Text(text = model.title, fontSize = 10.em,
                     modifier = Modifier.fillMaxWidth(), textAlign = TextAlign.Center)
                Image(
                    painter = rememberAsyncImagePainter(model.imageUrl),
                    contentDescription = null,
                    modifier = Modifier.fillMaxWidth(),
                    alignment = Alignment.Center
                )
            }
        }

    }

    data class MemeCardModel(val imageUrl: String?, val title: String)

...
```

* `implementation(„io.coil-kt:coil-compose:2.0.0-rc01")`

Android Basics

Exkurs: SwiftUI




```swift
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            Text("FCB vs. BVB").padding(.top, 10)

            AsyncImage(url: URL(string: "...")) { image in
                    image.resizable()
                } placeholder: {
                    ProgressView()
                }
                .frame(width: 200, height: 250)
                .background(Color.gray)
                .padding(10)
        }
        .background(Color(UIColor.lightGray))
        .cornerRadius(10)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```