STU FEI
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Mobilné výpočty

Ing. Maroš Čavojský, PhD.

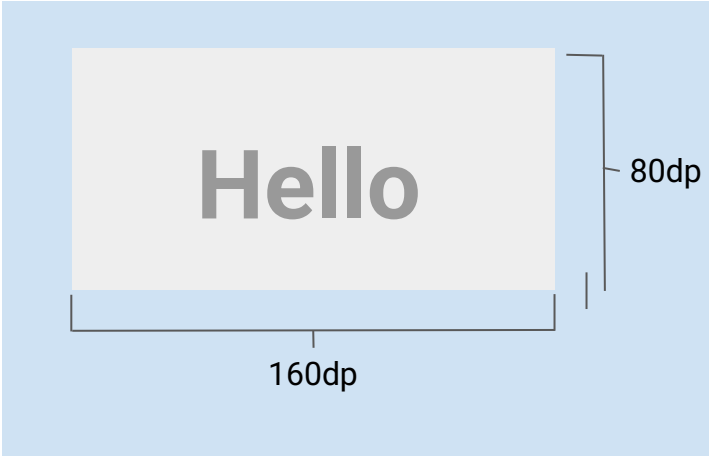
Android devices

- Android devices come in many different form factors.
- More and more pixels per inch are being packed into device screens.
- Developers need the ability to specify layout dimensions that are consistent across devices.

Density-independent pixels (dp)

Use dp when specifying sizes in your layout, such as the width or height of views.

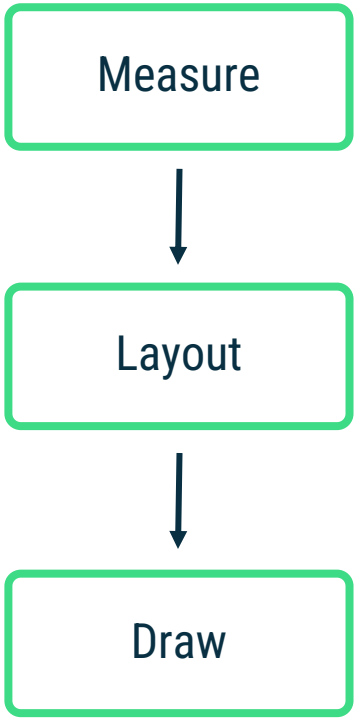
- Density-independent pixels (dp) take screen density into account.
- Android views are measured in density-independent pixels.
- dp = $\frac{\text{(width in pixels * 160)}}{\text{screen density}}$

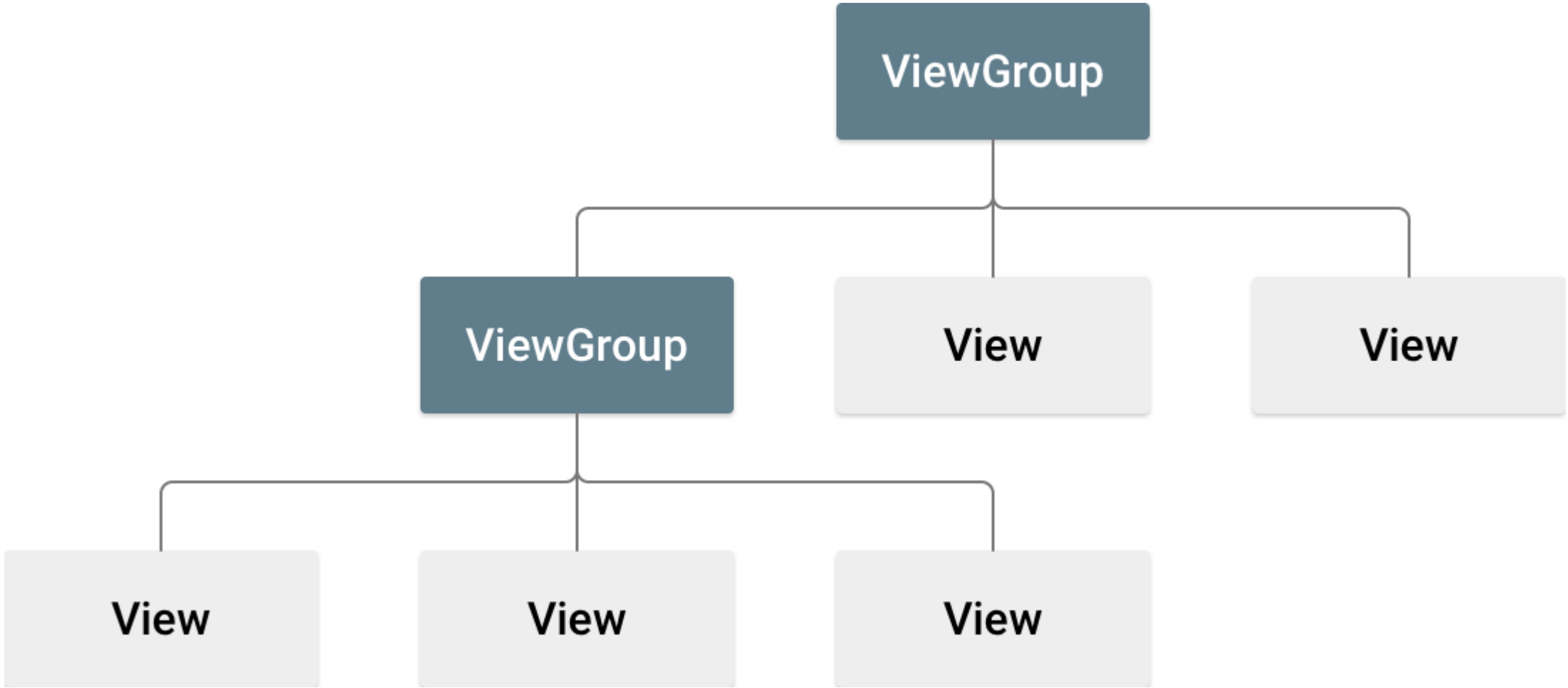


Screen-density buckets

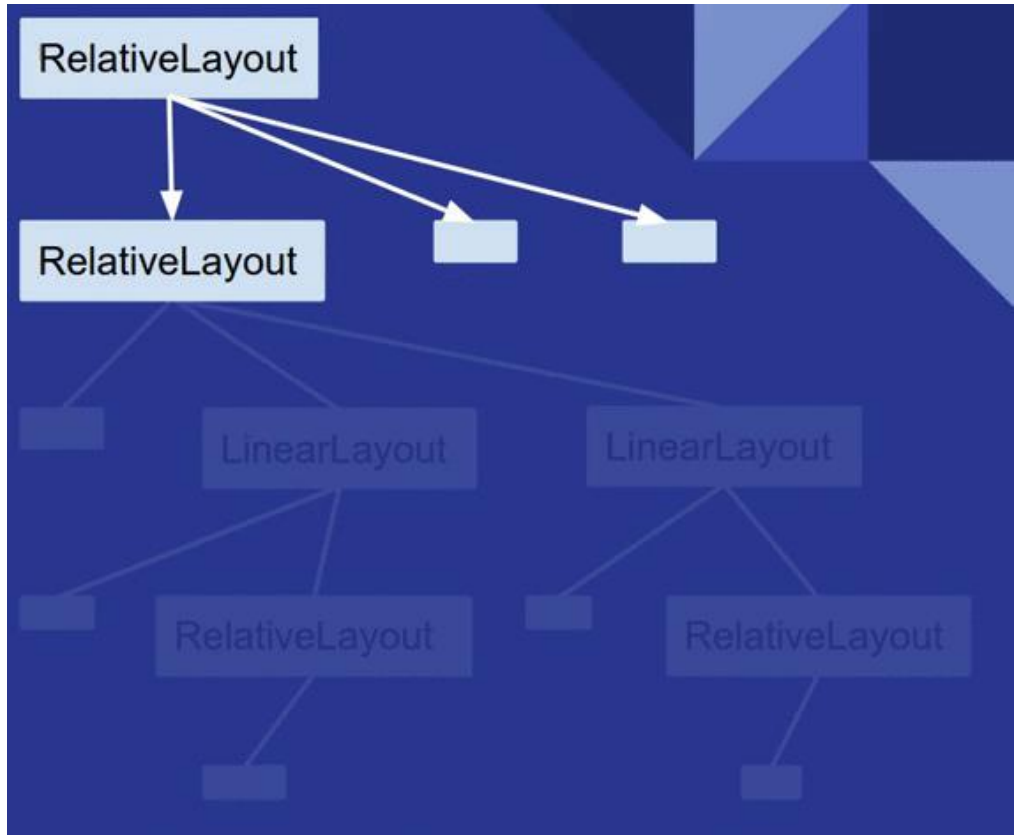
Density qualifier	Description	DPI estimate
ldpi (mostly unused)	Low density	~120dpi
mdpi (baseline density)	Medium density	~160dpi
hdpi	High density	~240dpi
xhdpi	Extra-high density	~320dpi
xxhdpi	Extra-extra-high density	~480dpi
xxxhdpi	Extra-extra-extra-high density	~640dpi

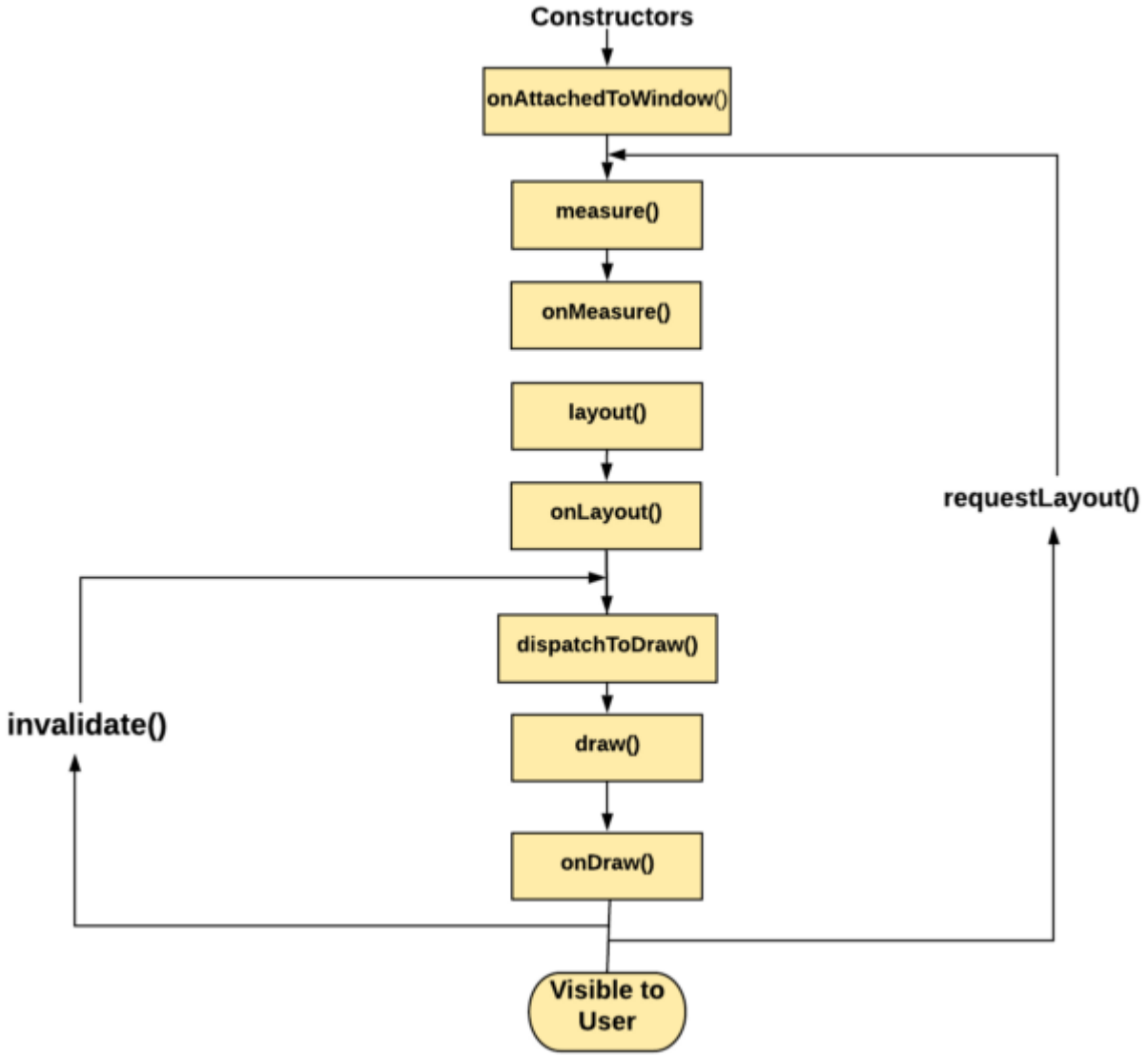
Android View rendering cycle



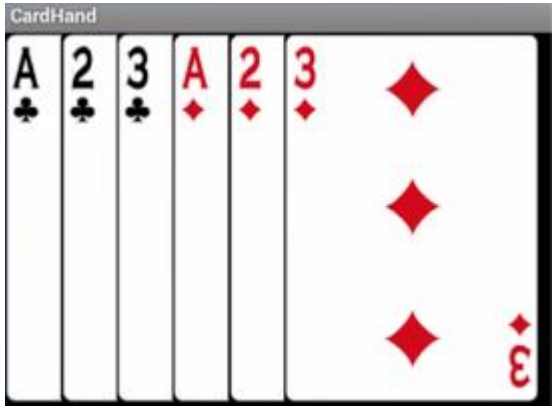

ViewGroup
ViewGroup
View
View
View
View
View

View / ViewGroup





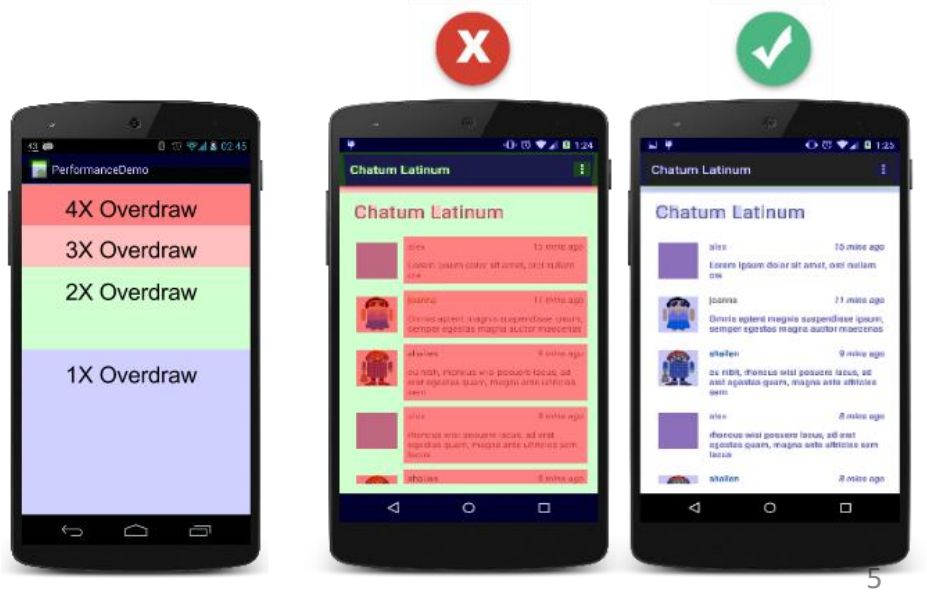
Chyby pri vytváraní UI



- Prekresľovanie – Overdraw
- Komplexné vnorené layouty – LinearLayout/RelativeLayout .... ConstraintLayout
- Double taxation – komplikované layouty vyžadujú viac-násobné layout-measure
  - RelativeLayout
  - LinearLayout
  - GridLayout

Vykreslenie za 16ms pre 60 fps

1000ms / 60 frames =
16.666 ms / frame



View

- základný stavebný prvok

ViewGroup

- podtrieda triedy View
- základný stavebný prvok pre rozloženia (neviditeľné prvky)

LinearLayout

- pre riadkové/stĺpcové rozloženie

ConstrainLayout

- pre efektívne rozloženie

RelativeLayout

- pre relatívne rozloženie

LinearLayout

- pre riadkové/stĺpcové rozloženie

Android Developers > Docs > Referencie

LinearLayout

```
open class LinearLayout : ViewGroup
```

kotlin.Any
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.LinearLayout

ConstrainLayout

- pre efektívne rozloženie

Android Developers > Docs > Reference

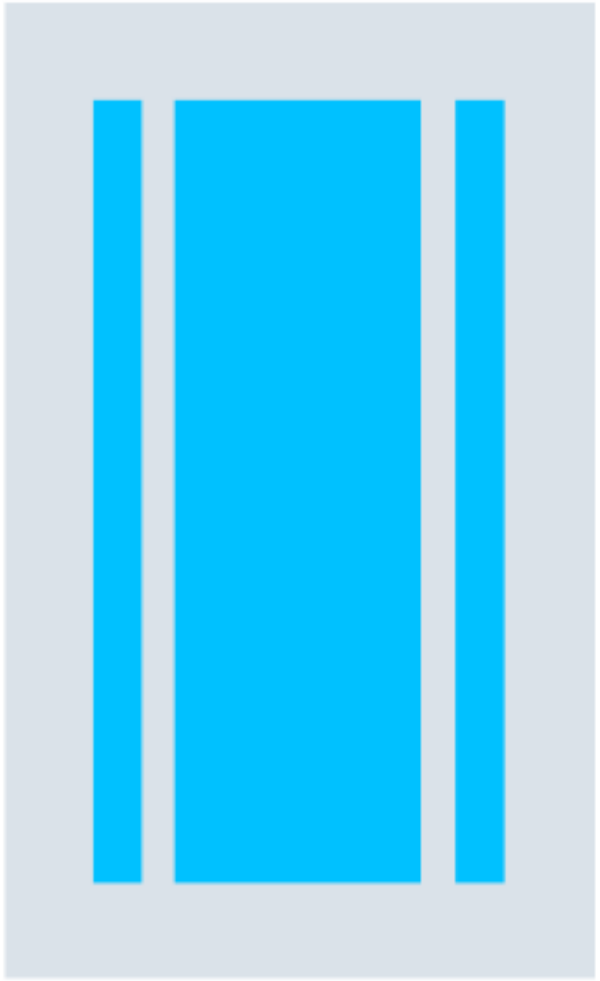
ConstraintLayout

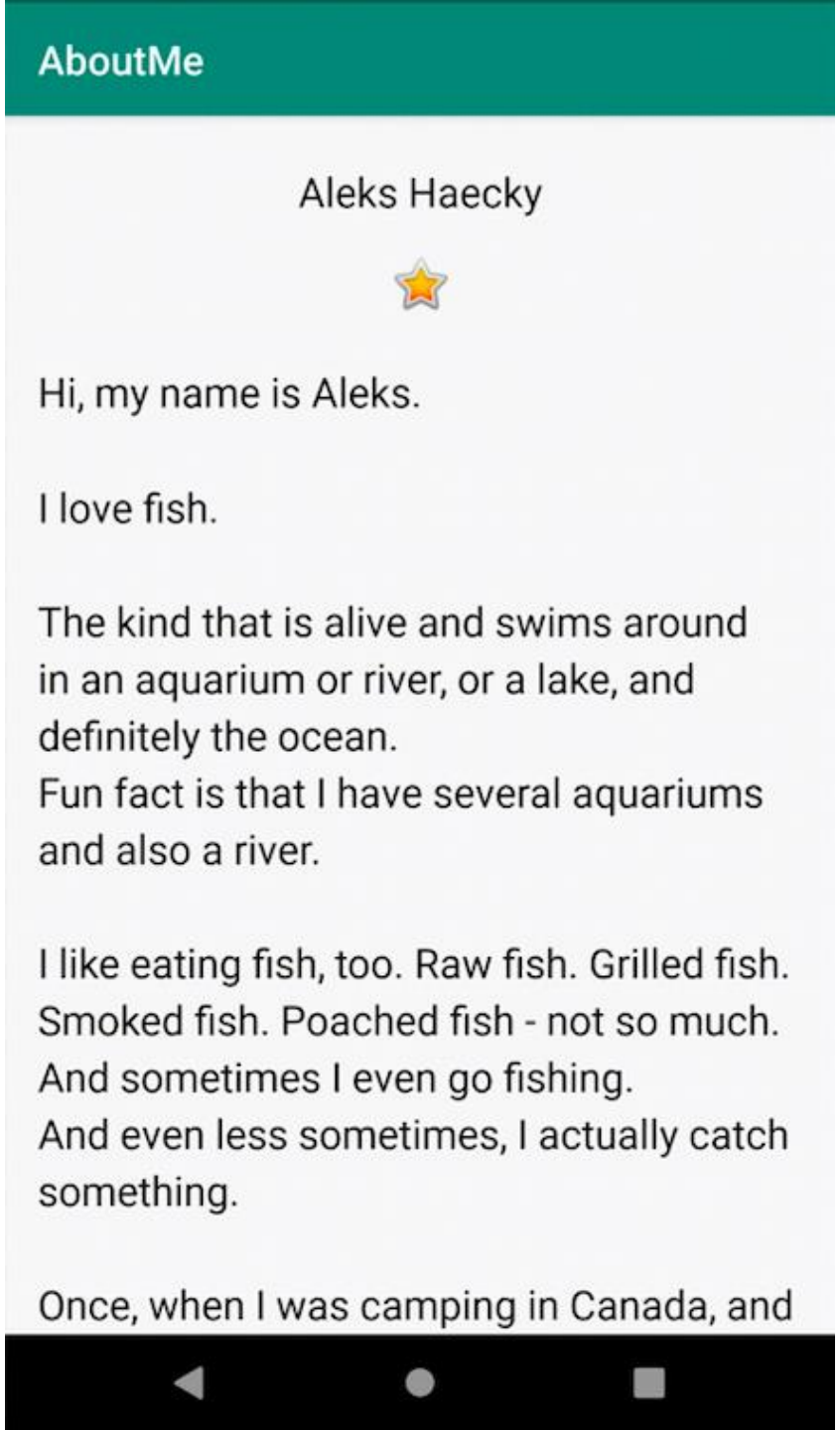
```
public class ConstraintLayout
extends ViewGroup
```

java.lang.Object
↳ ViewGroup
↳ androidx.constraintlayout.widget.ConstraintLayout

Vertical LinearLayout

Horizontal LinearLayout





Vertial LinearLayout

ScrollView

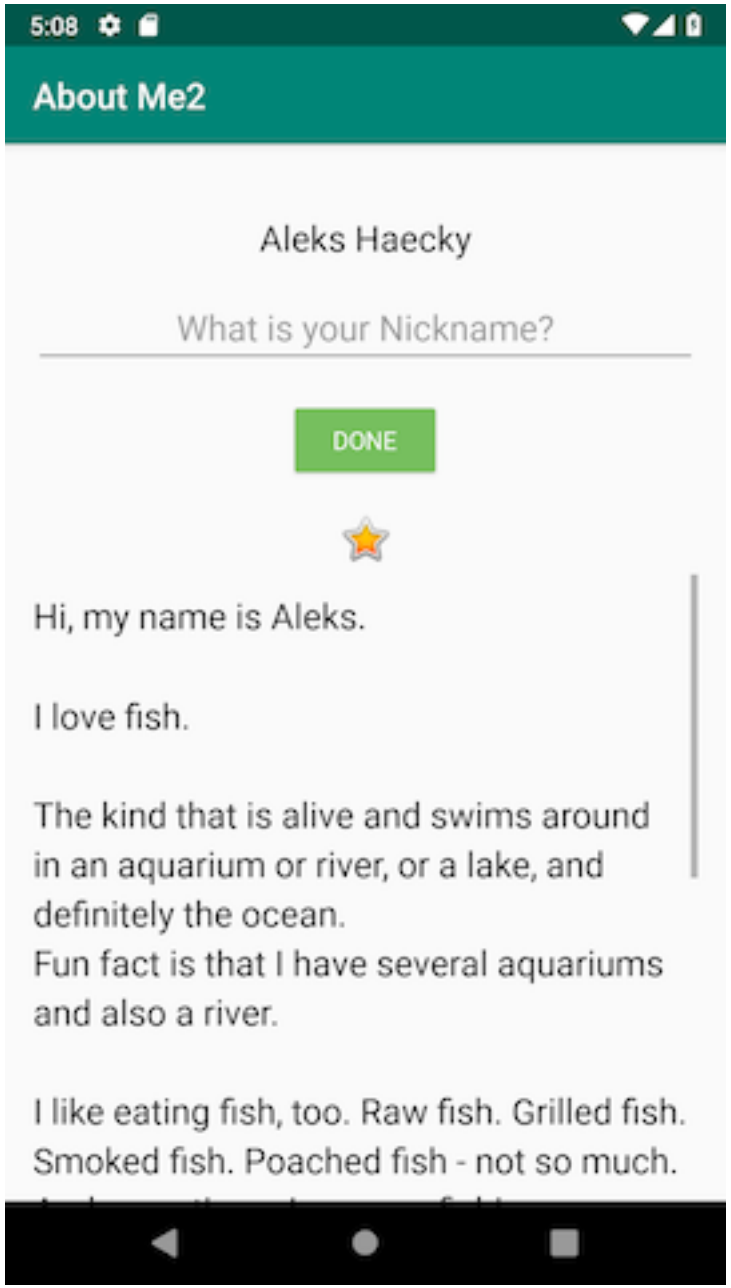
- umožňuje scrolovať obsah
- môže obsahovať najviac jeden prvok

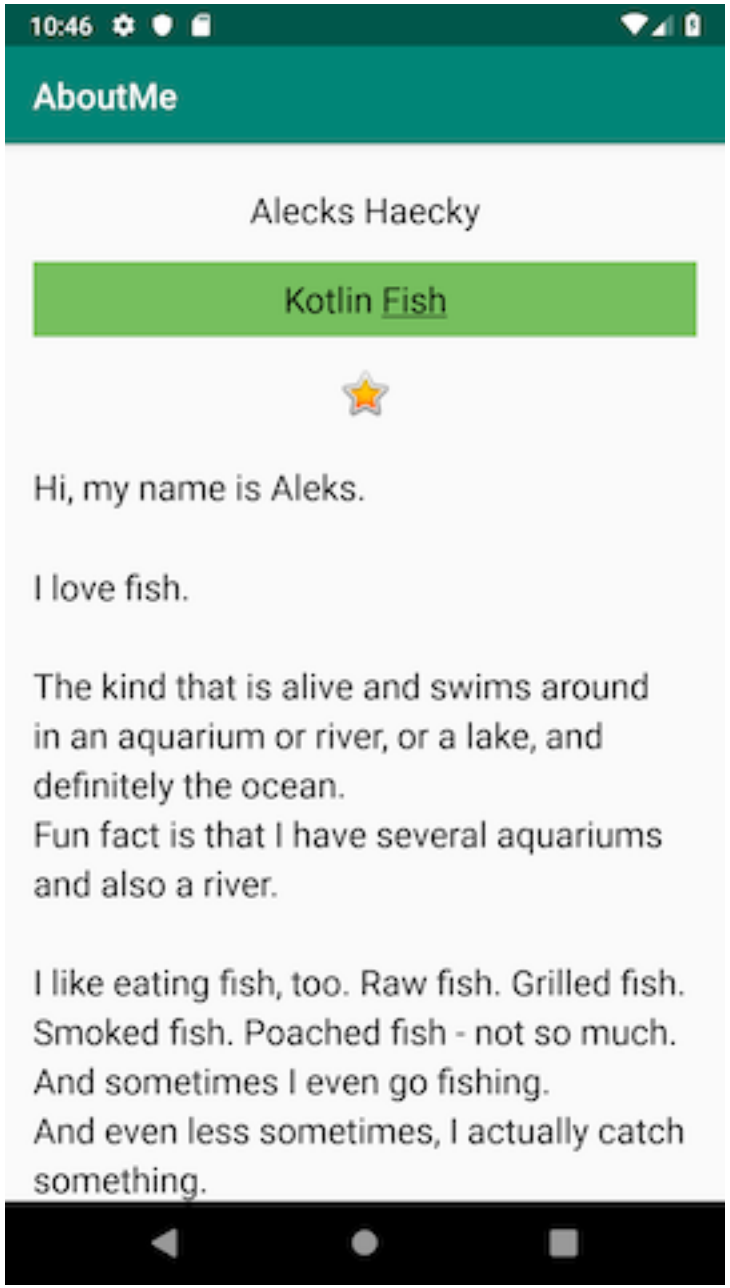
Android Developers > Docs > Referencie

ScrollView

```
open class ScrollView : FrameLayout
```

kotlin.Any
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.FrameLayout
↳ android.widget.ScrollView


5:08
About Me2
Aleks Haecky
What is your Nickname?
DONE
Hi, my name is Aleks.
I love fish.
The kind that is alive and swims around in an aquarium or river, or a lake, and definitely the ocean.
Fun fact is that I have several aquariums and also a river.
I like eating fish, too. Raw fish. Grilled fish. Smoked fish. Poached fish - not so much.


10:46
AboutMe
Alecks Haecky
Kotlin Fish
Hi, my name is Aleks.
I love fish.
The kind that is alive and swims around in an aquarium or river, or a lake, and definitely the ocean.
Fun fact is that I have several aquariums and also a river.
I like eating fish, too. Raw fish. Grilled fish. Smoked fish. Poached fish - not so much.
And sometimes I even go fishing.
And even less sometimes, I actually catch something.

EditText

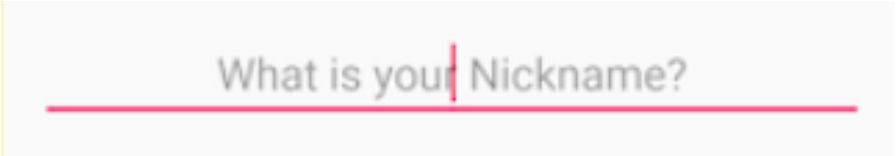
- umožňuje zadávať text

Android Developers > Docs > Referencie

EditText

```
open class EditText : TextView
```

kotlin.Any
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.EditText




```
<EditText
      android:id="@+id/plain_text_input"
      android:layout_height="wrap_content"
      android:layout_width="match_parent"
      android:inputType="text"/>
```

TextView

- umožňuje zobrazovať text

```
<TextView
    android:id="@+id/text_view_id"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/hello" />
```

Android Developers > Docs > Referencie

TextView

```
open class TextView : View, ViewTreeObserver.OnPreDrawListener
```

kotlin.Any
↳ android.view.View
↳ android.widget.TextView

Button

- umožňuje potvrdiť / vykonať akciu

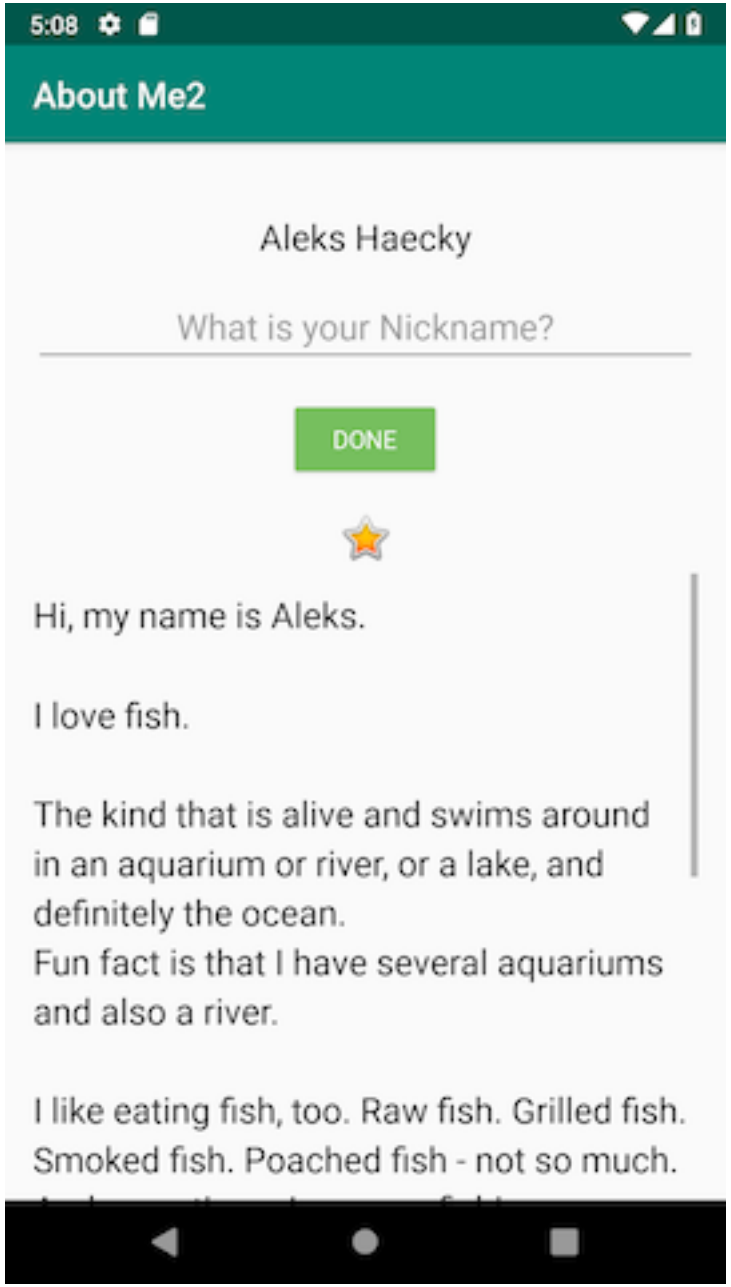
Android Developers > Docs > Referencie

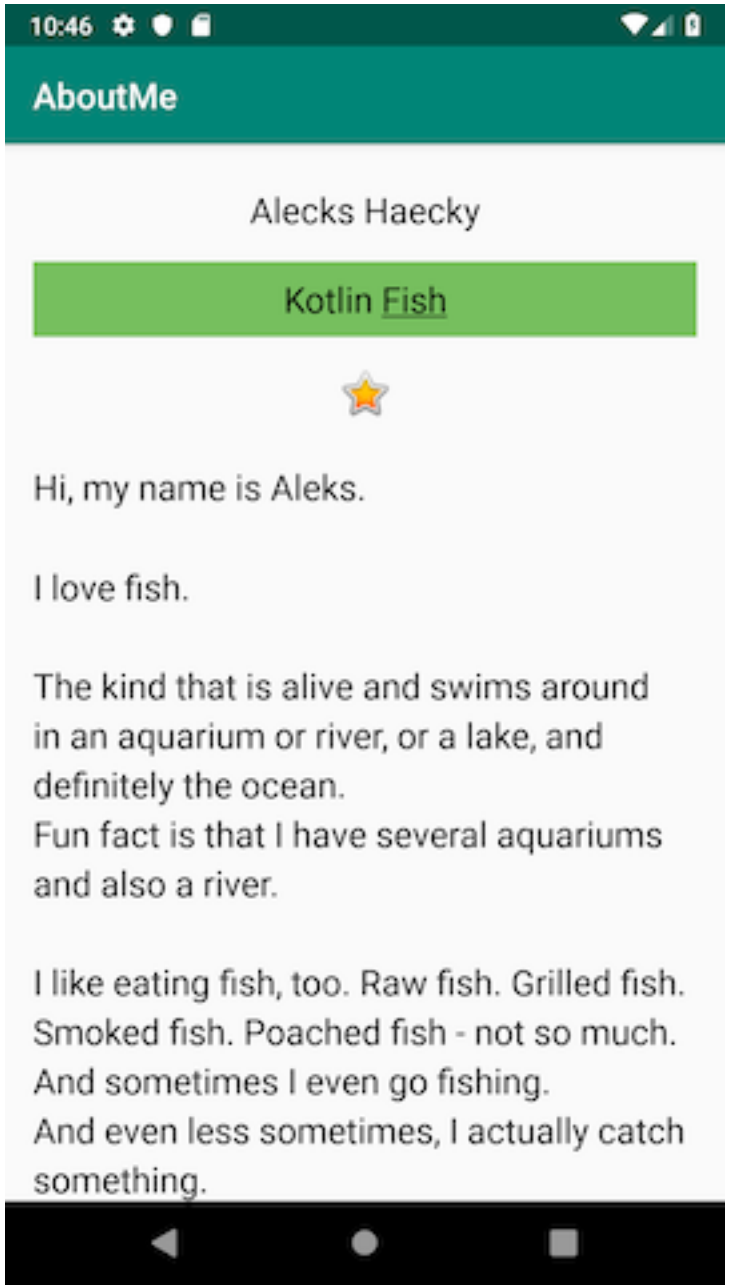
Button

```
open class Button : TextView
```

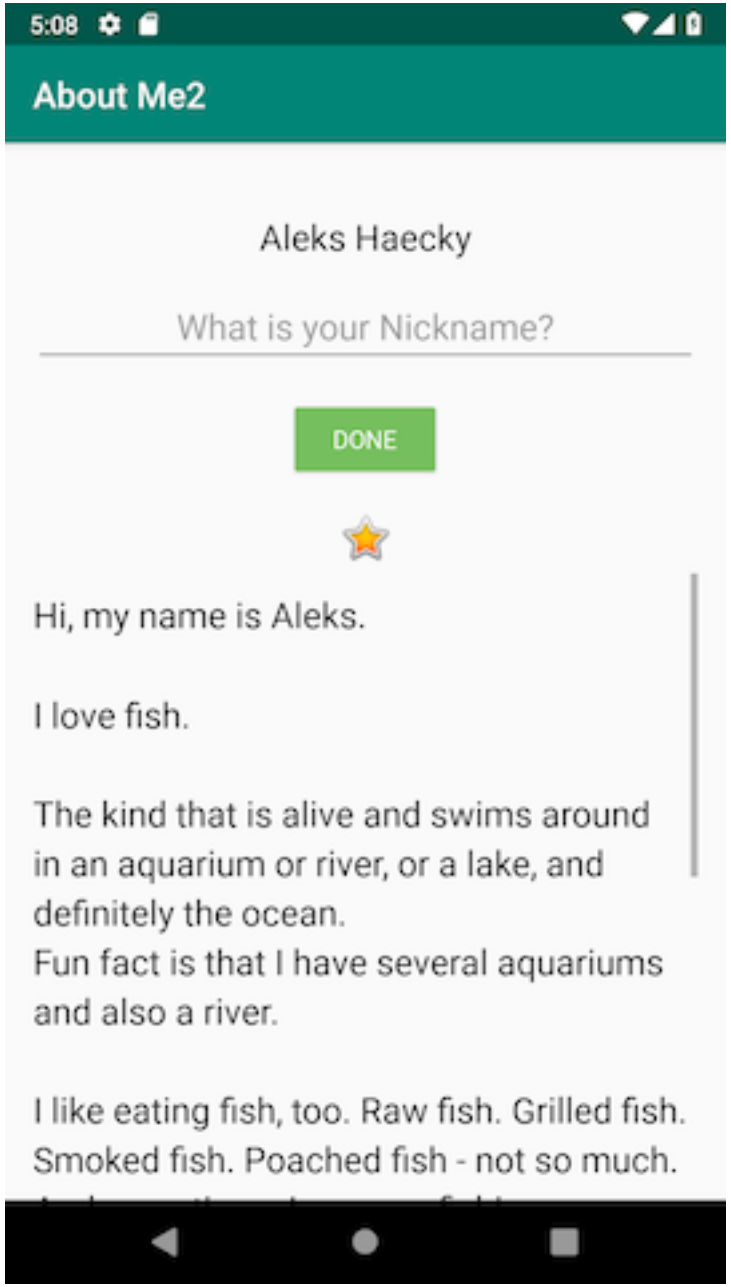
kotlin.Any
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.Button

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


5:08
About Me2
Aleks Haecky
What is your Nickname?
DONE
Hi, my name is Aleks.
I love fish.
The kind that is alive and swims around in an aquarium or river, or a lake, and definitely the ocean.
Fun fact is that I have several aquariums and also a river.
I like eating fish, too. Raw fish. Grilled fish. Smoked fish. Poached fish - not so much.

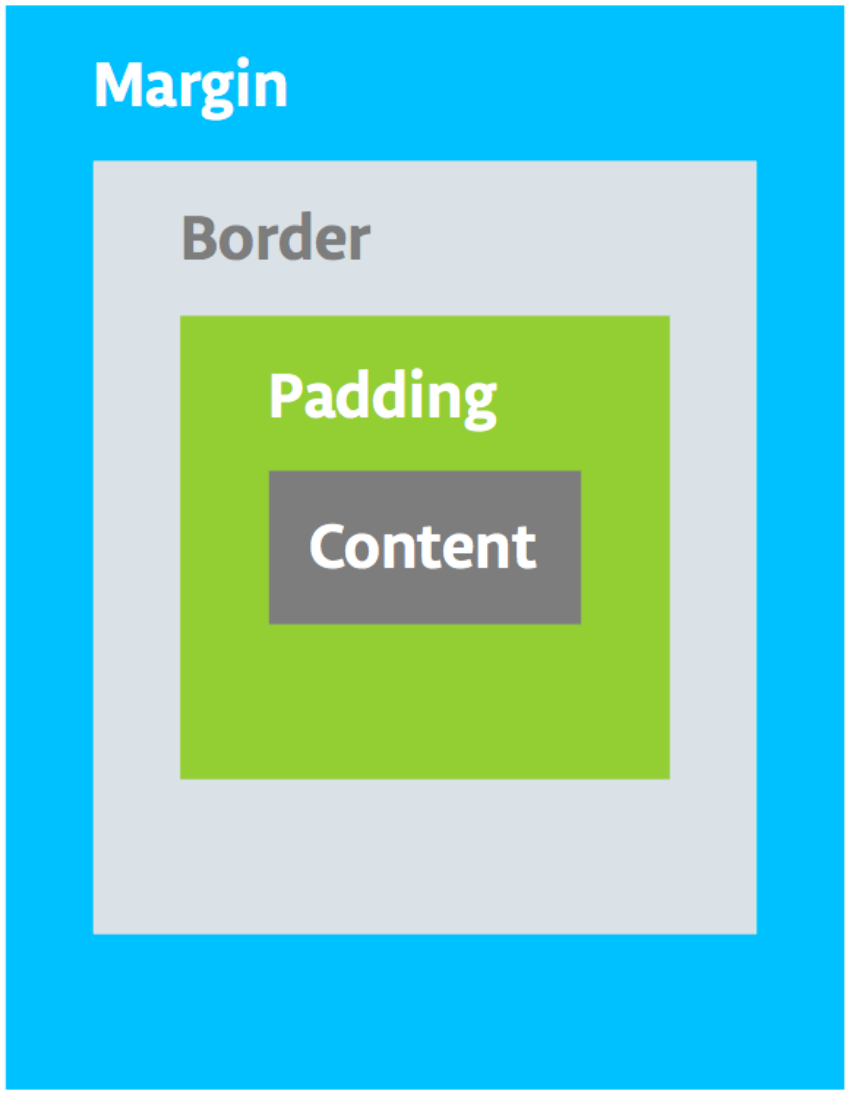

10:46
AboutMe
Alecks Haecky
Kotlin Fish
Hi, my name is Aleks.
I love fish.
The kind that is alive and swims around in an aquarium or river, or a lake, and definitely the ocean.
Fun fact is that I have several aquariums and also a river.
I like eating fish, too. Raw fish. Grilled fish. Smoked fish. Poached fish - not so much. And sometimes I even go fishing.
And even less sometimes, I actually catch something.

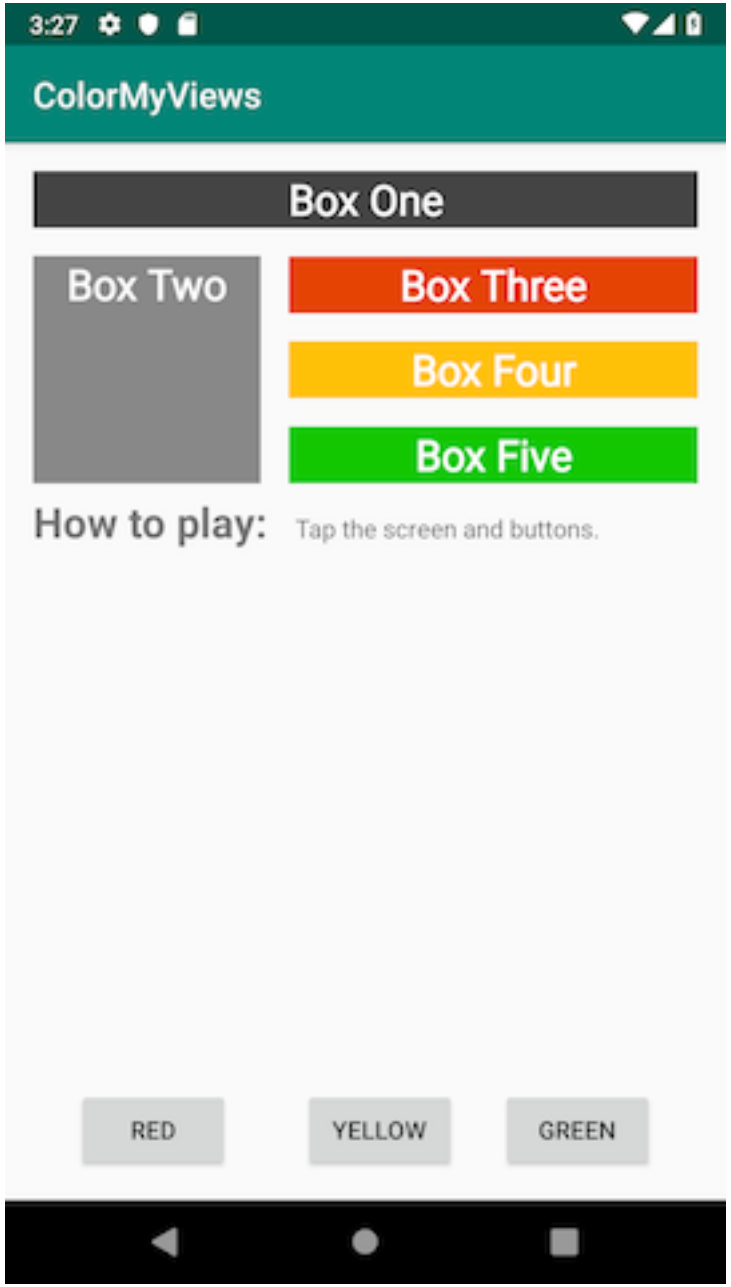

ScrollView
LinearLayout
TextView
EditText
TextView
Button
ImageView
TextView
visible
scroll to




```
android:visibility="gone"
```

- visible = viditeľné
- invisible = neviditeľné (zaberá miesto)
- gone = neviditeľné (nezaberá miesto)


Margin
Border
Padding
Content



ConstraintLayout

- pre efektívne rozloženie

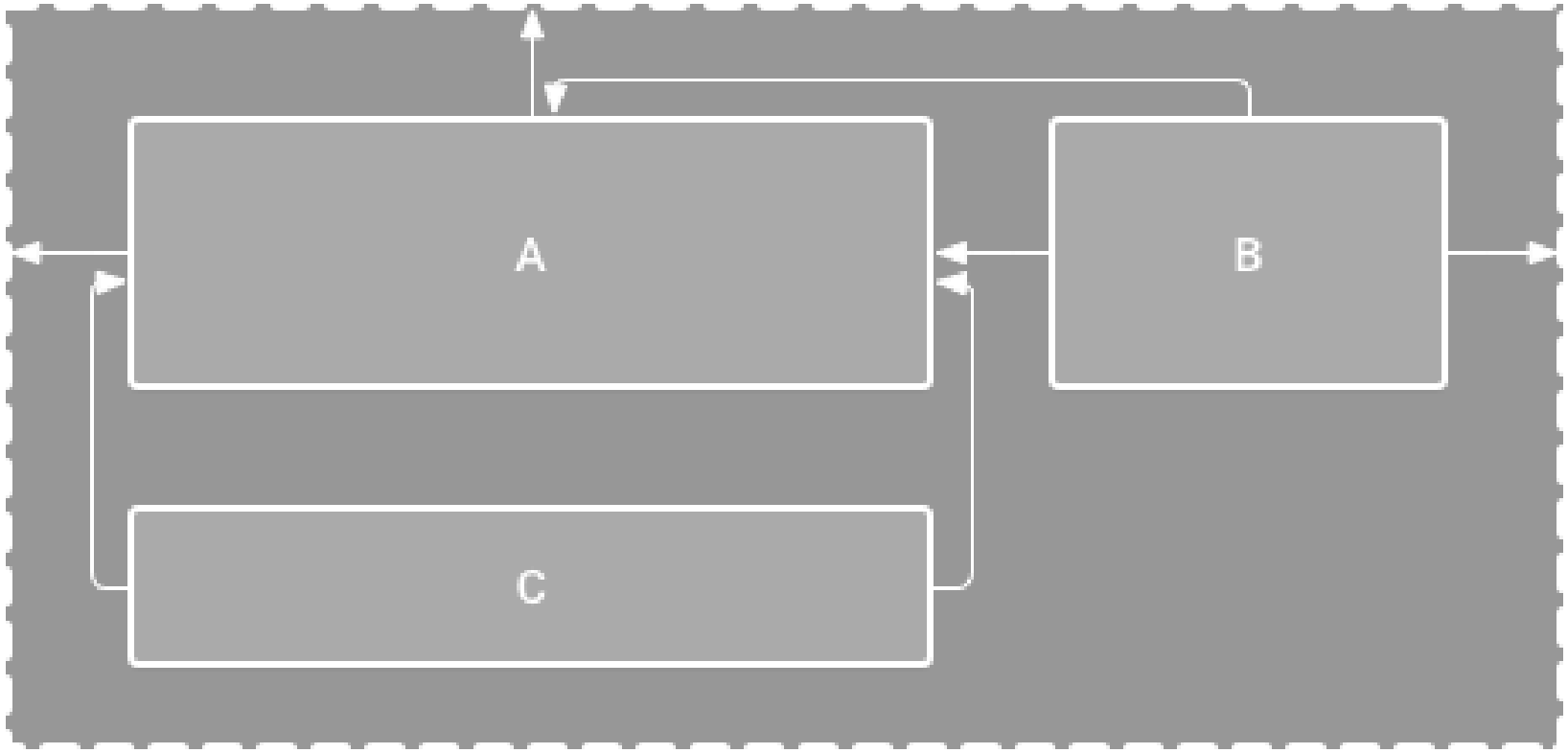
Android Developers > Docs > Reference

ConstraintLayout

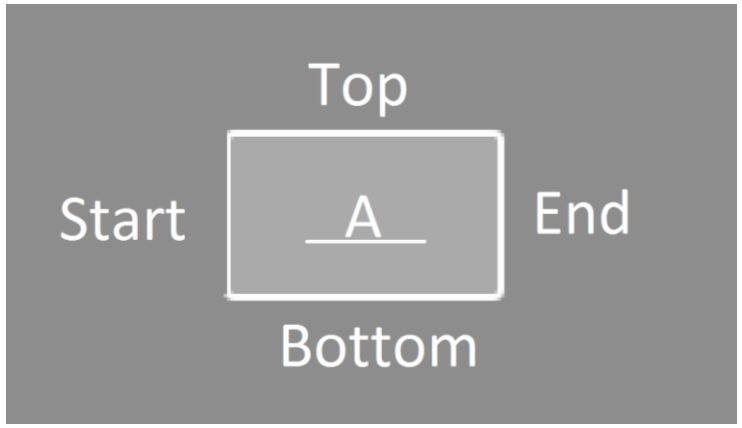
```
public class ConstraintLayout
extends ViewGroup
```

java.lang.Object
↳ ViewGroup
↳ androidx.constraintlayout.widget.ConstraintLayout

ConstraintLayout



- layout_constraintLeft_toLeftOf
- layout_constraintLeft_toRightOf
- layout_constraintRight_toLeftOf
- layout_constraintRight_toRightOf



- layout_constraintStart_toEndOf
- layout_constraintStart_toStartOf

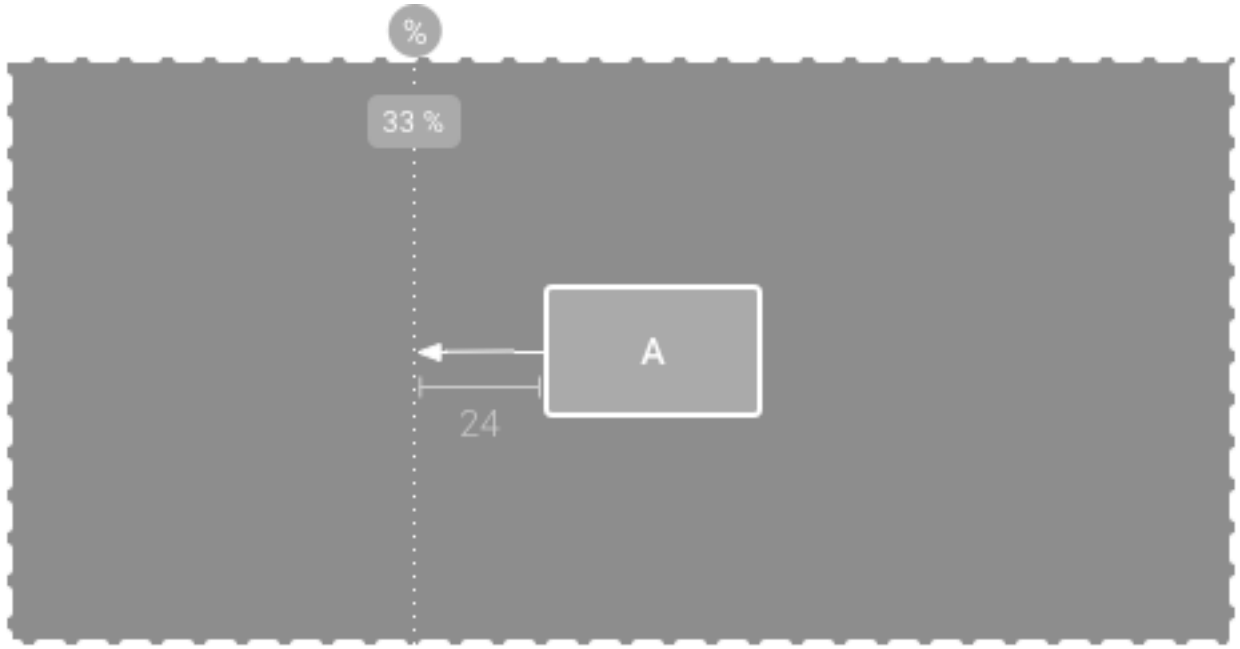
- layout_constraintEnd_toStartOf
- layout_constraintEnd_toEndOf

- layout_constraintTop_toTopOf
- layout_constraintTop_toBottomOf

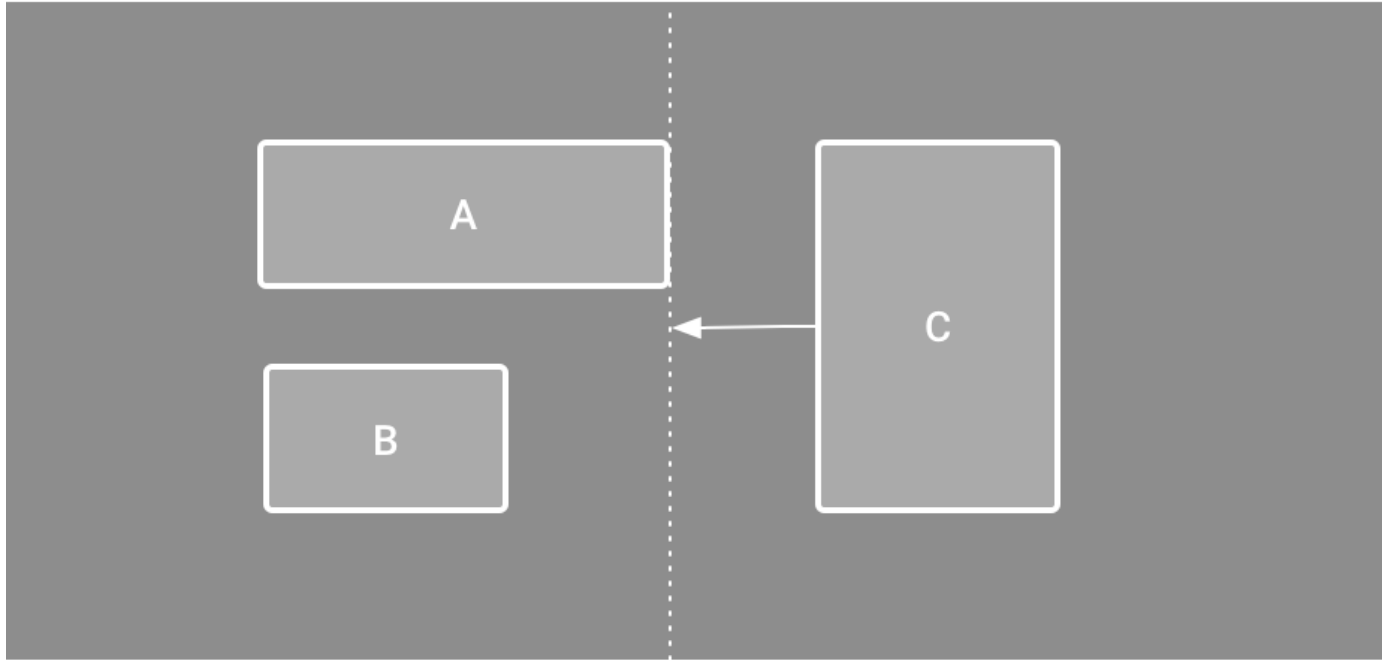
- layout_constraintBottom_toTopOf
- layout_constraintBottom_toBottomOf

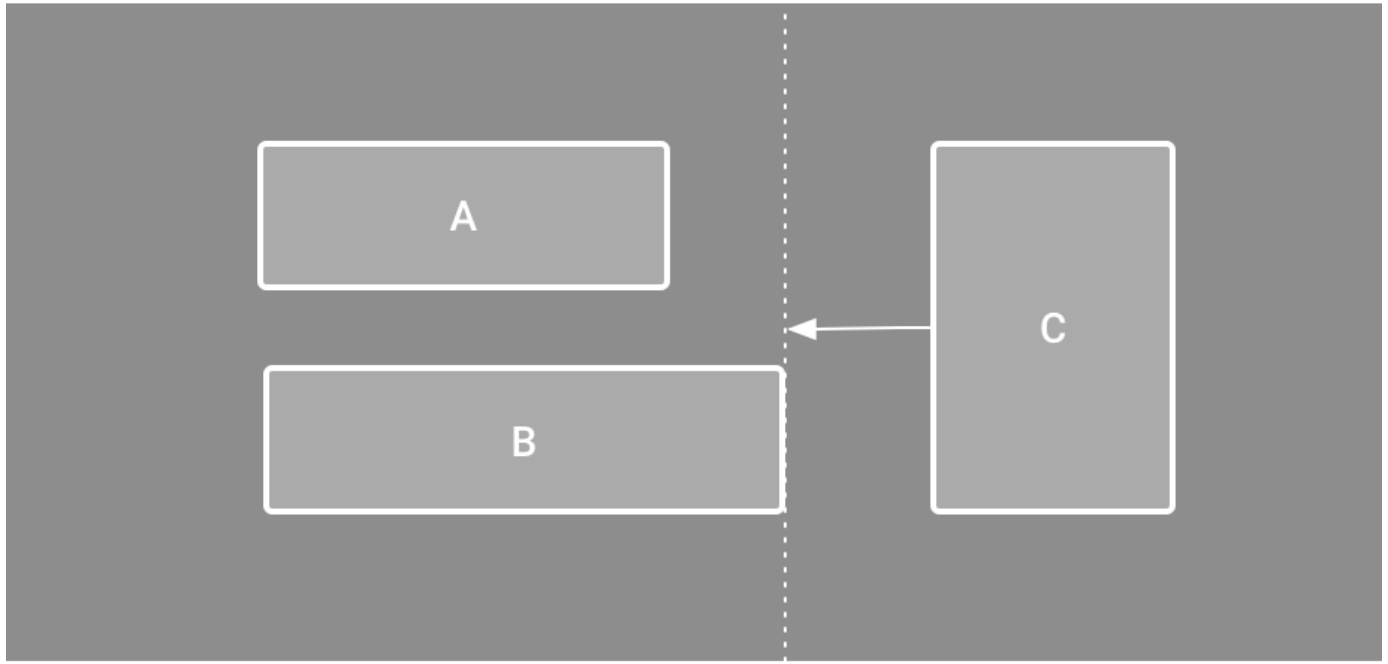
- layout_constraintBaseline_toBaselineOf

Guideline

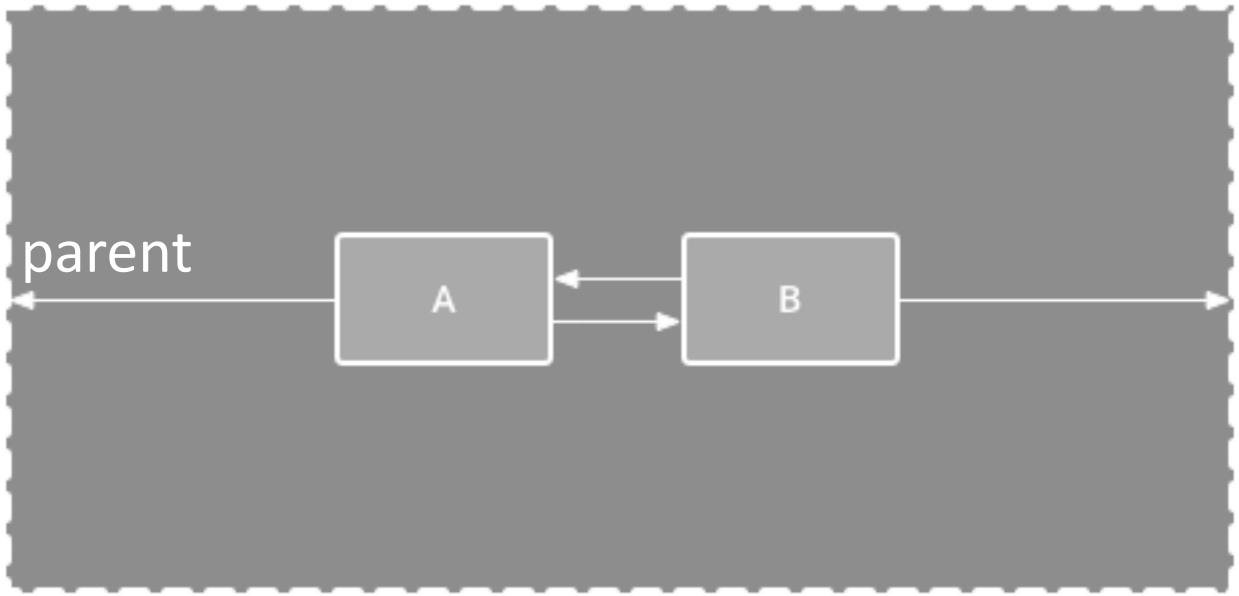


Barier

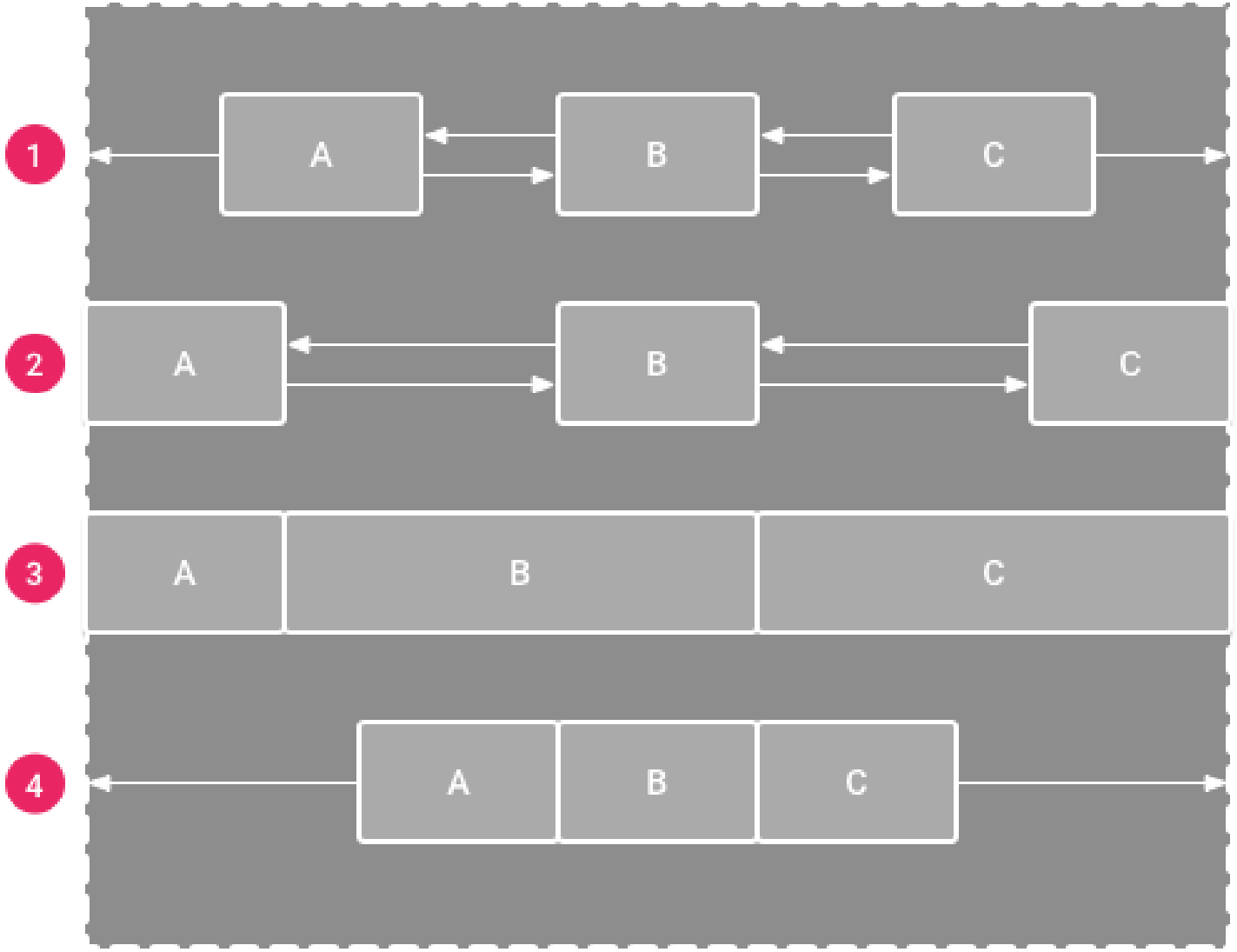




Chain



1. SPREAD
   - rovnomerne rozložené
   - predvolený štýl
2. SPREAD INSIDE
   - prvý a posledný prilepené
   - stredné rovnomerne rozložené
3. WEIGHTED
   - SPREAD alebo SPREAD INSIDE
   - nastaviť váhy pre šírku
4. PACKED
   - prilepené k sebe



Intenty

- Implicitné

```
val webIntent: Intent = Uri.parse("http://www.android.com").let { webpage ->
    Intent(Intent.ACTION_VIEW, webpage)
}
```

```
val callIntent: Intent = Uri.parse("tel:5551234").let { number ->
    Intent(Intent.ACTION_DIAL, number)
}
```

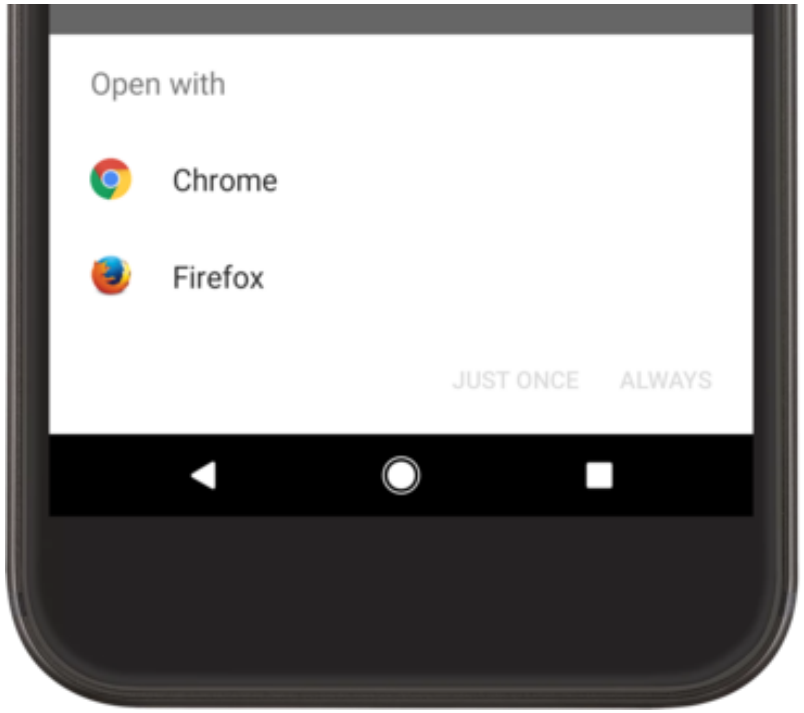
- Explicitné

```
val explicitIntent: Intent = Intent(context,MojaActivita::class.java)
```

Intenty – implicitné

```
// Build the intent
val location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California")
val mapIntent = Intent(Intent.ACTION_VIEW, location)

// Verify it resolves
val activities: List<ResolveInfo> = packageManager.queryIntentActivities(mapIntent, 0)
val isIntentSafe: Boolean = activities.isNotEmpty()

// Start an activity if it's safe
if (isIntentSafe) {
    startActivity(mapIntent)
}
```

Intenty (komunikácia)

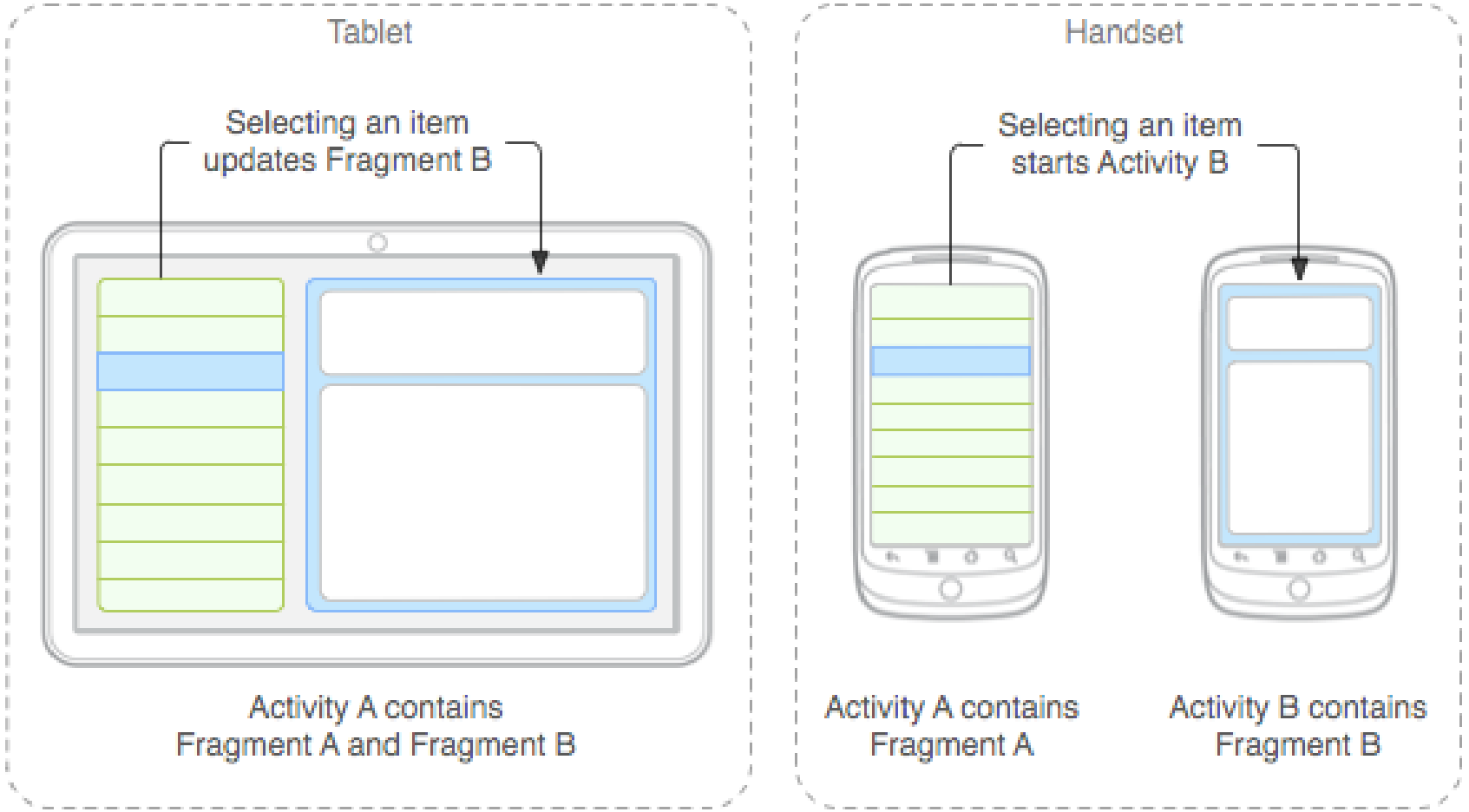


Intenty (komunikácia)

```
val intent = Intent(Intent.ACTION_SEND)
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
val title = resources.getString(R.string.chooser_title)
// Create intent to show chooser
val chooser = Intent.createChooser(intent, title)

// Verify the intent will resolve to at least one activity
if (intent.resolveActivity(packageManager) != null) {
    startActivity(chooser)
}
```

Fragmenty

Fragmenty


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

Fragmenty

```
class ExampleFragment : Fragment() {

    override fun onCreateView(
            inflater: LayoutInflater,
            container: ViewGroup?,
            savedInstanceState: Bundle?
    ): View {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment, container, false)
    }
}
```

Fragmenty

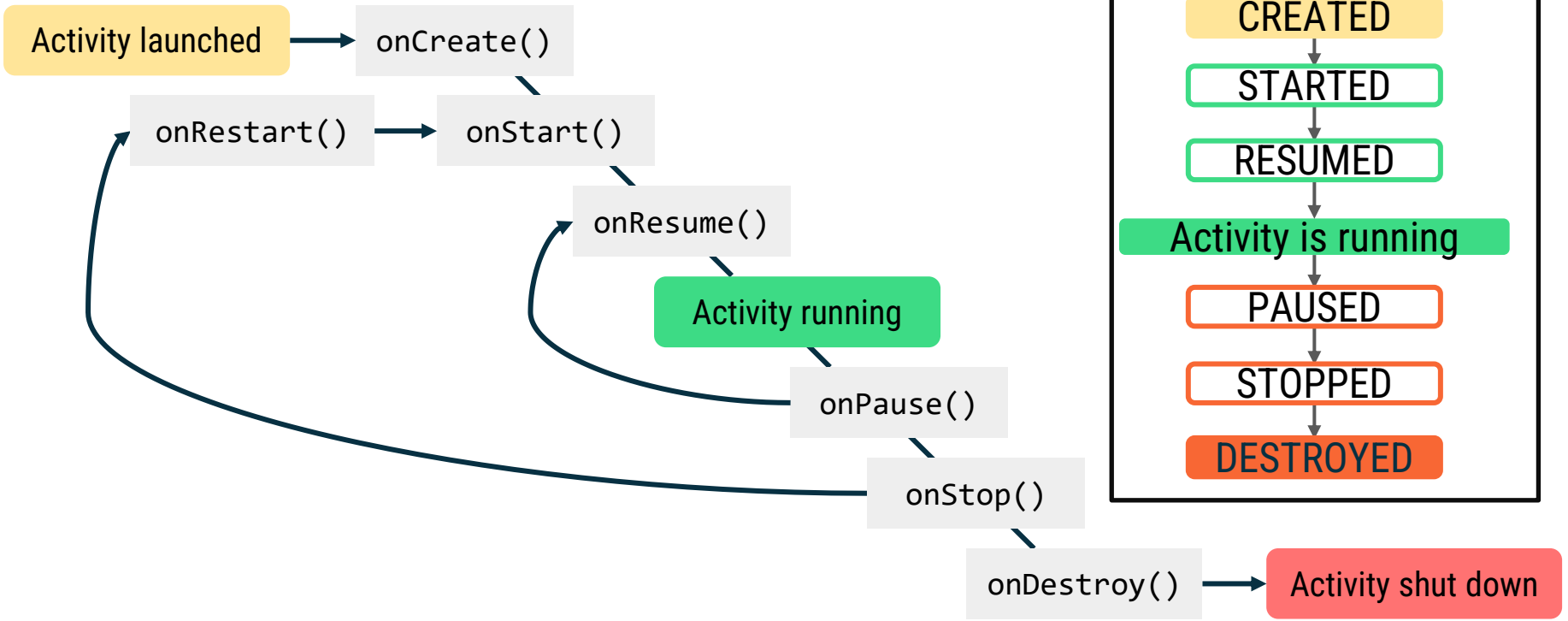
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

Fragmenty

```
val fragmentManager = supportFragmentManager
val fragmentTransaction = fragmentManager.beginTransaction()
```

```
val fragment = ExampleFragment()
fragmentTransaction.add(R.id.fragment_container, fragment)
fragmentTransaction.commit()
```

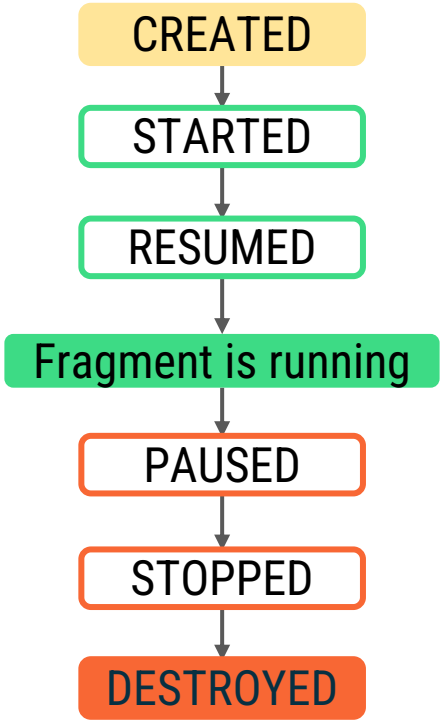
Activity lifecycle



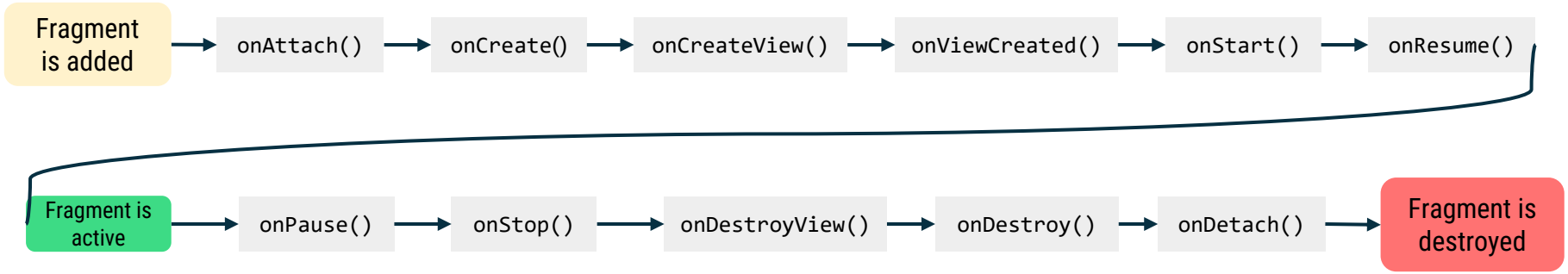
Summary of activity states

State	Callbacks	Description
Created	`onCreate()`	Activity is being initialized.
Started	`onStart()`	Activity is visible to the user.
Resumed	`onResume()`	Activity has input focus.
Paused	`onPause()`	Activity does not have input focus.
Stopped	`onStop()`	Activity is no longer visible.
Destroyed	`onDestroy()`	Activity is destroyed.

Fragment states



Fragment lifecycle diagram



Summary of fragment states

State	Callbacks	Description
Initialized	`onAttach()`	Fragment is attached to host.
Created	`onCreate()`, `onCreateView()`, `onViewCreated()`	Fragment is created and layout is being initialized.
Started	`onStart()`	Fragment is started and visible.
Resumed	`onResume()`	Fragment has input focus.
Paused	`onPause()`	Fragment no longer has input focus.
Stopped	`onStop()`	Fragment is not visible.
Destroyed	`onDestroyView()`, `onDestroy()`, `onDetach()`	Fragment is removed from host.

Navigation component

- Collection of libraries and tooling, including an integrated editor, for creating navigation paths through an app
- Assumes one `Activity` per graph with many `Fragment` destinations
- Consists of three major parts:
  - Navigation graph
  - Navigation Host (`NavHost`)
  - Navigation Controller (`NavController`)

Add dependencies

In `build.gradle.kts`, under `dependencies`:

```
implementation("androidx.navigation:navigation-fragment-ktx:$nav_version")
implementation("androidx.navigation:navigation-ui-ktx:$nav_version")
```

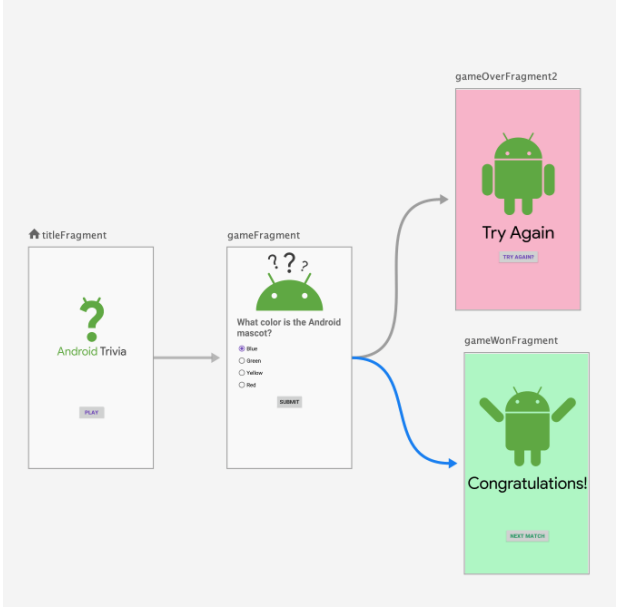
Navigation host (NavHost)

```
<fragment
    android:id="@+id/nav_host"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph_name"/>
```

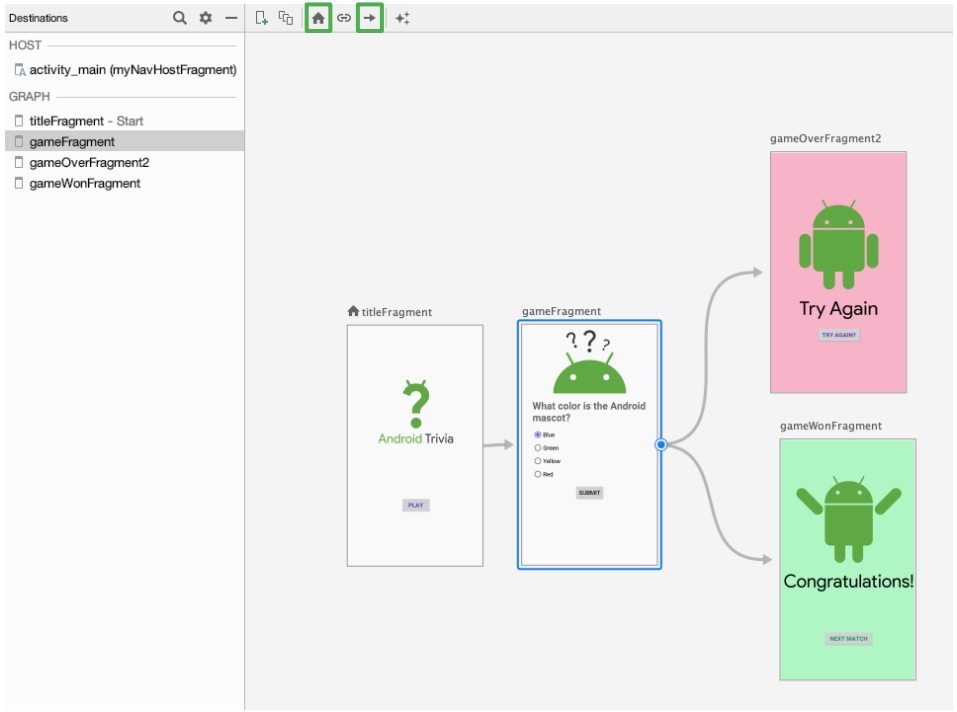
Navigation graph

New resource type located in `res/navigation` directory

- XML file containing all of your navigation destinations and actions
- Lists all the (Fragment/Activity) destinations that can be navigated to
- Lists the associated actions to traverse between them
- Optionally lists animations for entering or exiting



Navigation Editor in Android Studio



Specifying Fragment destinations

- Fragment destinations are denoted by the `action` tag in the navigation graph.
- Actions can be defined in XML directly or in the Navigation Editor by dragging from source to destination.
- Autogenerated action IDs take the form of `action_<sourceFragment>_to_<destinationFragment>`.

Example fragment destination

```
<fragment
    android:id="@+id/welcomeFragment"
    android:name="com.example.android.navigation.WelcomeFragment"
    android:label="fragment_welcome"
    tools:layout="@layout/fragment_welcome" >

    <action
        android:id="@+id/action_welcomeFragment_to_detailFragment"
        app:destination="@id/detailFragment" />

</fragment>
```

Navigation Controller (NavController)

`NavController` manages UI navigation in a navigation host.

- Specifying a destination path only names the action, but it doesn't execute it.
- To follow a path, use `NavController`.

Example NavController

```
myButton.setOnClickListener { view ->
    view.findNavController().navigate(R.id.someDestination)
}
```

Vo Fragmente:

```
val navController = findNavController()
```

V Aktivite:

```
val navController = findNavController(R.id.nav_host_fragment)
```

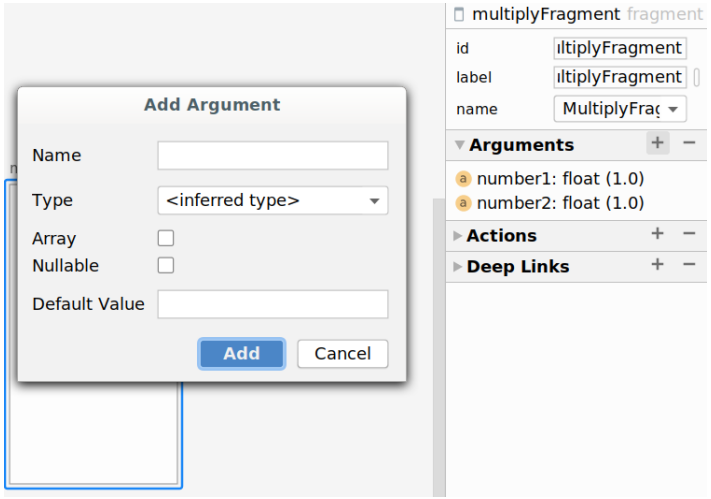
Passing data between destinations

Using Safe Args:

- Ensures arguments have a valid type
- Lets you provide default values
- Generates a `<SourceDestination>Directions` class with methods for every action in that destination
- Generates a class to set arguments for every named action
- Generates a `<TargetDestination>Args` class providing access to the destination's arguments

Destination arguments

```
<fragment
    android:id="@+id/multiplyFragment"
    android:name="com.example.arithmetic.MultiplyFragment"
    android:label="MultiplyFragment" >
    <argument
        android:name="number1"
        app:argType="float"
        android:defaultValue="1.0" />
    <argument
        android:name="number2"
        app:argType="float"
        android:defaultValue="1.0" />
 </fragment>
```



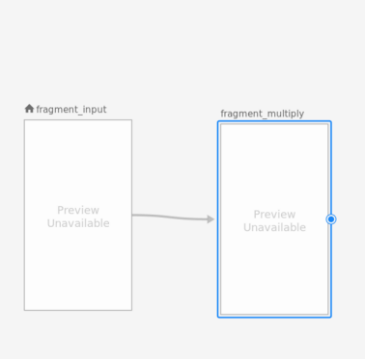
Supported argument types

Type	**Type Syntax** `app:argType=<type>`	**Supports Default Values**	**Supports Null Values**
Integer	`"integer"`	Yes	No
Float	`"float"`	Yes	No
Long	`"long"`	Yes	No
Boolean	`"boolean"`	Yes (`"true"` or `"false"`)	No
String	`"string"`	Yes	Yes
Array	above type + `"[]"` (for example, `"string[]"` `"long[]"`)	Yes (only `"@null"`)	Yes
Enum	Fully qualified name of the enum	Yes	No
Resource reference	`"reference"`	Yes	No

Create action from source to destination

In `nav_graph.xml`:

```
<fragment
    android:id="@+id/fragment_input"
    android:name="com.example.arithmetic.InputFragment">
    <action
        android:id="@+id/action_to_multiplyFragment"
        app:destination="@id/fragment_multiply" />
</fragment>
<fragment
    android:id="@+id/fragment_multiply"
    android:name="com.example.arithmetic.MultiplyFragment">
```



Navigating with actions

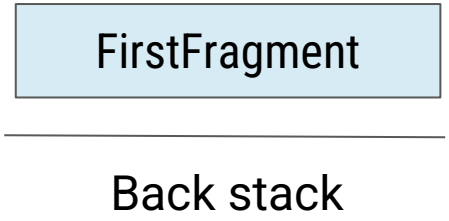
In `InputFragment.kt`:

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    binding.button.setOnClickListener {
        val n1 = binding.number1.text.toString().toFloatOrNull() ?: 0.0
        val n2 = binding.number2.text.toString().toFloatOrNull() ?: 0.0

        val action = InputFragmentDirections.actionToMultiplyFragment(n1, n2)
        view.findNavController().navigate(action)
    }
}
```

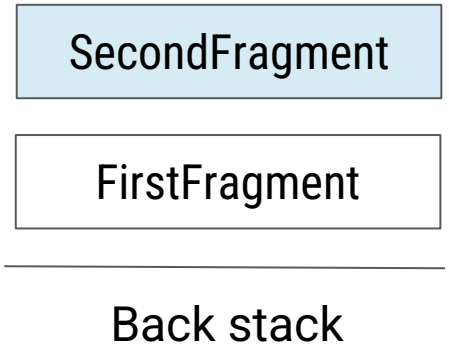
Retrieving Fragment arguments

```
class MultiplyFragment : Fragment() {
    val args: MultiplyFragmentArgs by navArgs()
    lateinit var binding: FragmentMultiplyBinding
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val number1 = args.number1
        val number2 = args.number2
        val result = number1 * number2
        binding.output.text = "${number1} * ${number2} = ${result}"
    }
}
```

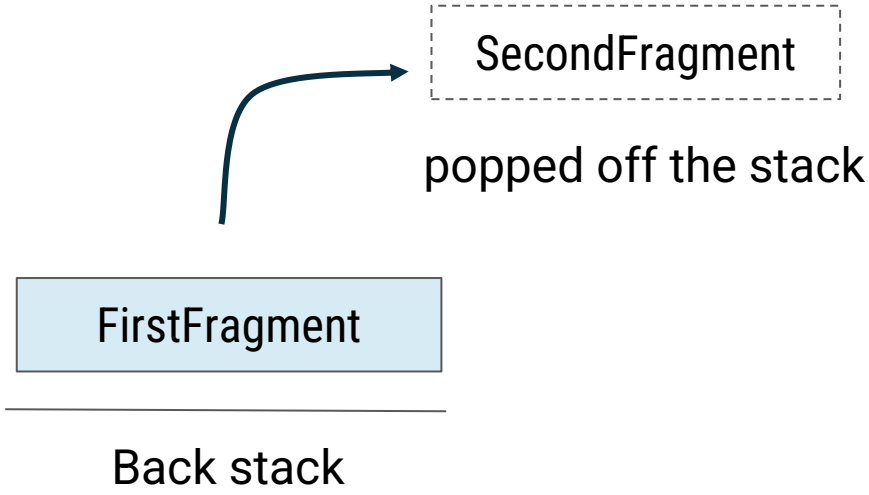
First destination in the back stack



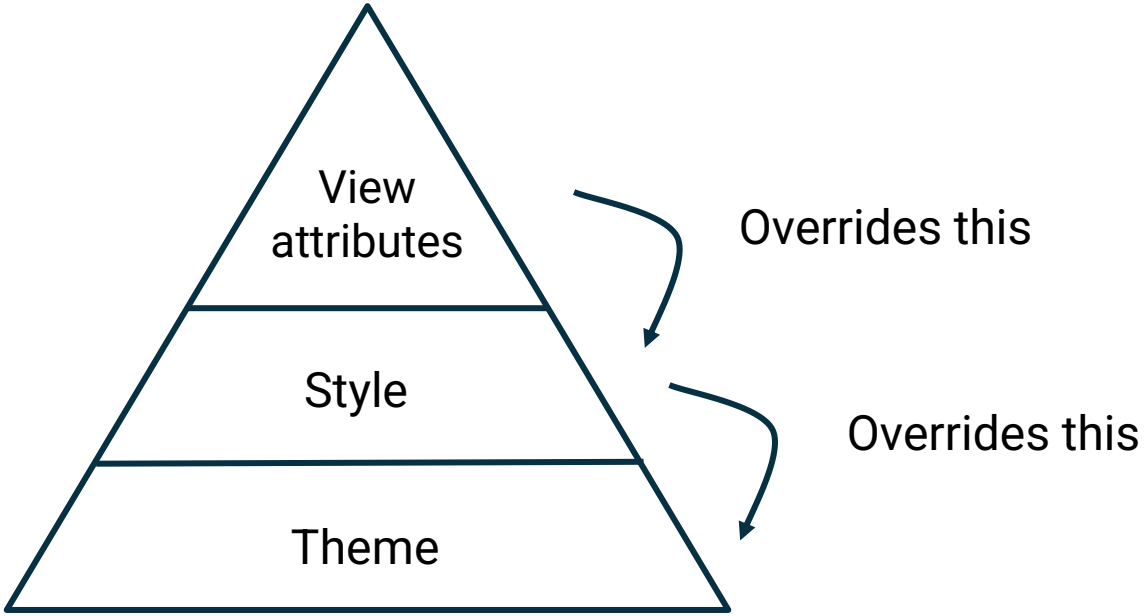
Add a destination to the back stack



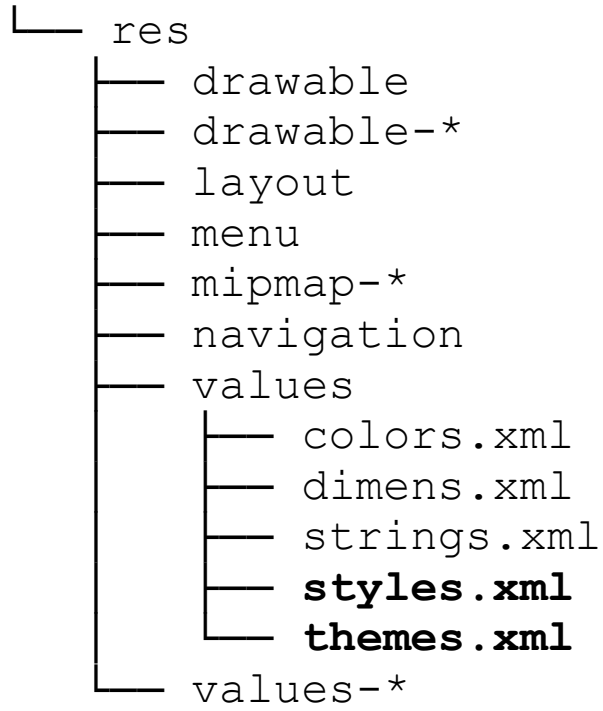
Tap Back button



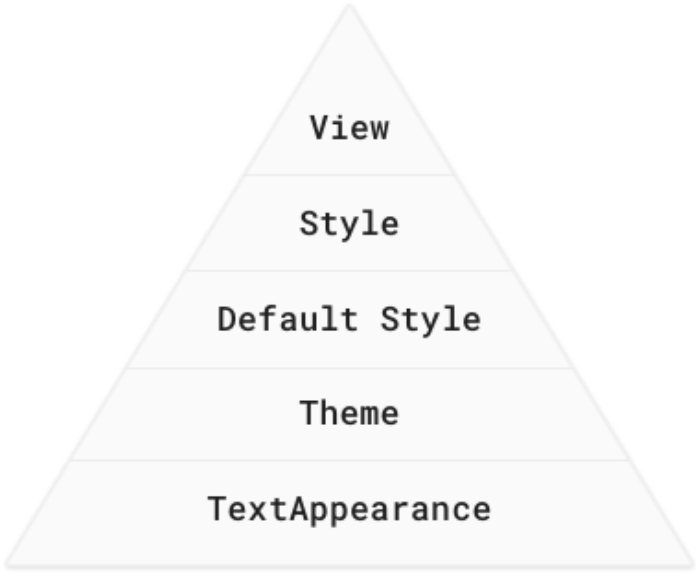
Precedence of each method of styling



Resources directory

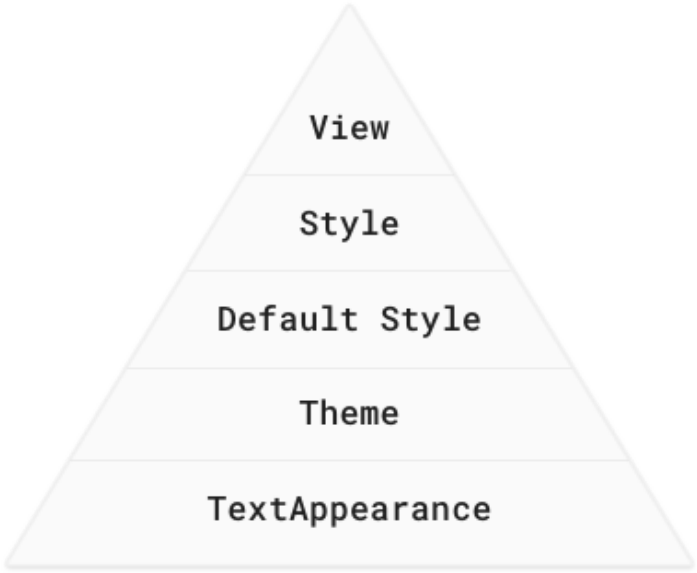


View




```
<Button
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="12sp“
    android:textColor="#008577"
.../>
```

View

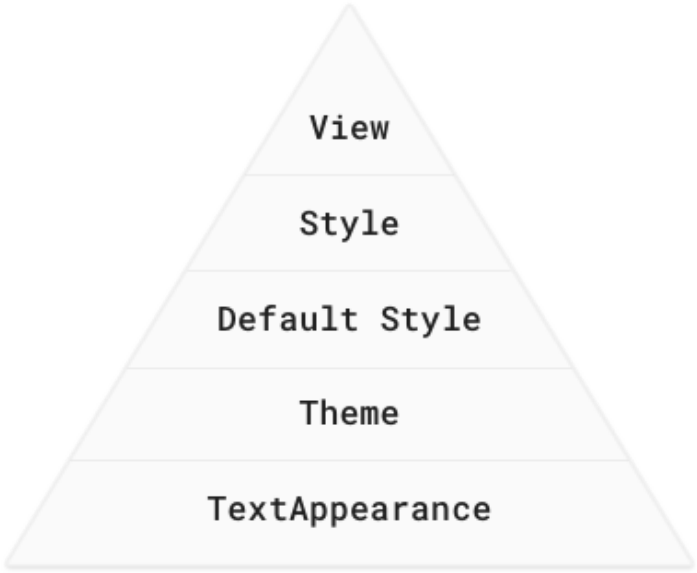



```
<Button
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="@dimen/textMedium"
    android:textColor="@color/colorPrimary"
...>
```

```
<dimen name="textSmall">11sp</dimen>
<dimen name="textMedium">12sp</dimen>
<dimen name="textBig">14sp</dimen>
```

```
<color name="colorPrimary">#008577</color>
<color name="colorPrimaryDark">#00574B</color>
<color name="colorAccent">#D81B60</color>
```

Style




```
<Button
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/Text"
...>
```

```
<style name="Text" parent="TextAppearance.AppCompat">
    <item name="android:textColor">@color/colorPrimary</item>
    <item name="android:textSize">@dimen/textMedium</item>
</style>
```

Theme

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


```
<application
        android:theme="@style/AppTheme.Launcher"
      ... >
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>

            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
</application>
```

Apply a theme

In `AndroidManifest.xml`:

```
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.MyApp" ... >
        </activity>
    </application>
</manifest>
```

In layout file:

```
<ConstraintLayout …
    android:theme="@style/Theme.MyApp">
```

Refer to theme attribute in a layout

In layout file:

```
<LinearLayout …
    android:background="?attr/colorSurface">
```

Use `?attr/themeAttributeName` syntax.

Examples: `?attr/colorPrimary`
`?attr/colorPrimaryVariant`

Declare a style

In `res/values/styles.xml`:

```
<style name="DescriptionStyle">
    <item name="android:textColor">#00FF00</item>
    <item name="android:textSize">16sp</item>
    ...
</style>
```

Apply a style

On a view in a layout file:

```
<TextView
    style="@style/DescriptionStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/description_text" />
```

Refer to theme attribute in a style

In `res/values/styles.xml`:

```
<style name="DescriptionStyle">
    <item name="android:textColor">?attr/colorOnSurface</item>
    <item name="android:textSize">16sp</item>
    ...
</style>
```

Color resources

A way to name and standardize colors throughout your app

In `res/values/colors.xml`:

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    ...
</resources>
```

Specified as hexadecimal colors in form of `#AARRGGBB`

Dimension resources

A way to name and standardize dimension values in your layouts

- Declare your dimension values in `res/values/dimens.xml`:

```
<resources>
    <dimen name="top_margin">16dp</dimen>
</resources>
```

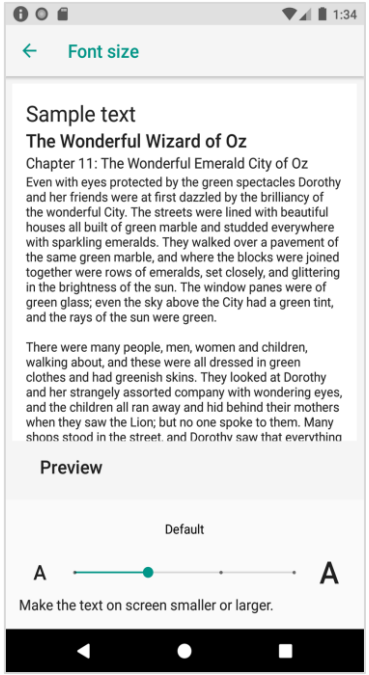
- Refer to them as `@dimen/<name>` in layouts or `R.dimen.<name>` in code:

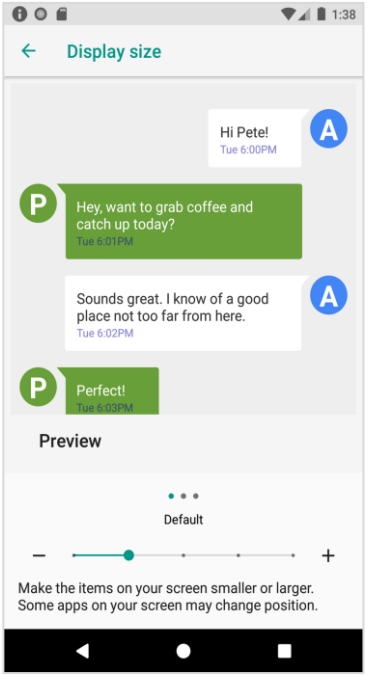
```
<TextView …
    android:layout_marginTop="@dimen/top_margin" />
```

Typography

Scale-independent pixels (sp)

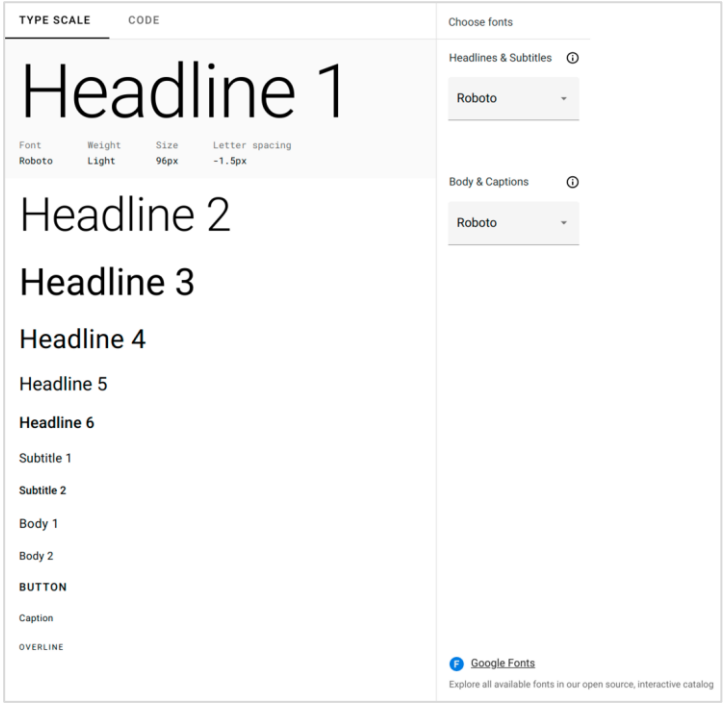
- The textual equivalent to density-independent pixels (dp)
- Specify text sizes in sp
  (takes into account user preferences)
- Users can adjust Font and Display sizes in the Settings app (after Display)





Type scale

- A set of styles designed to work together in a cohesive manner for your app and content
- Contains reusable categories of text with intended purpose for each (for example, headline, subtitle, caption)



TextAppearance

A `TextAppearance` style often alters one or more of these attributes:

- typeface (`android:fontFamily`)
- weight (`android:textStyle`)
- text size (`android:textSize`)
- capitalization (`android:textAllCaps`)
- letter spacing (`android:letterSpacing`)

Examples using TextAppearance

```
<TextView
    ...
    android:textAppearance="@style/TextAppearance.MaterialComponents.Headline1"
    android:text="@string/title" />


<TextView
    ...
    android:textAppearance="@style/TextAppearance.MaterialComponents.Body1"
    android:text="@string/body_text" />
```

Customize your own TextAppearance

```
<style name="TextAppearance.MyApp.Headline1"
        parent="TextAppearance.MaterialComponents.Headline1">
    ...
    <item name="android:textStyle">normal</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textSize">64sp</item>
    <item name="android:letterSpacing">0</item>
    ...
</style>
```

Use a custom TextAppearance in a theme

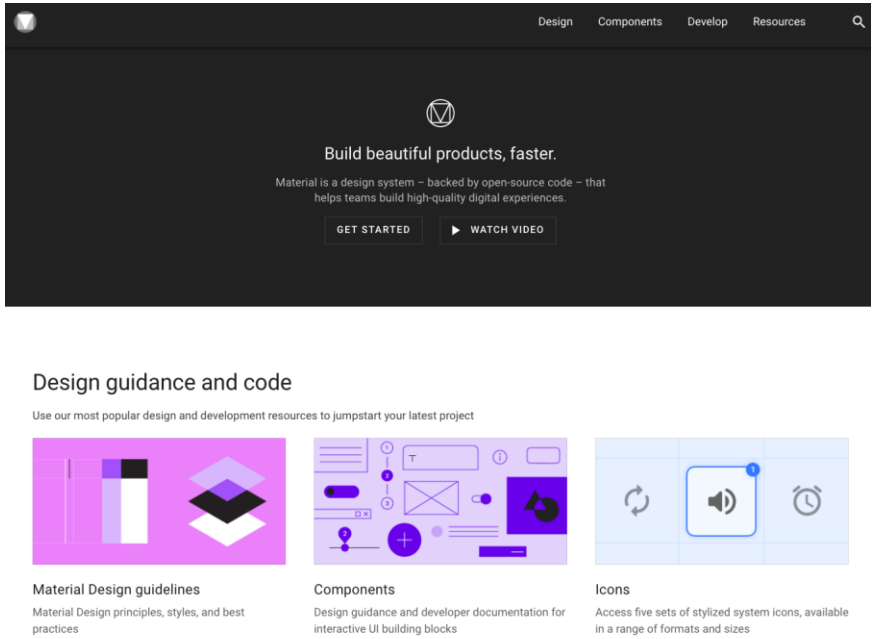
```
<style name="Theme.MyApp" parent="Theme.MaterialComponents.Light">
    ...
    <item name="textAppearanceHeadline1">@style/TextAppearance.MyApp.Headline1</item>
    ...
</style>
```

Material Design

Intro to Material

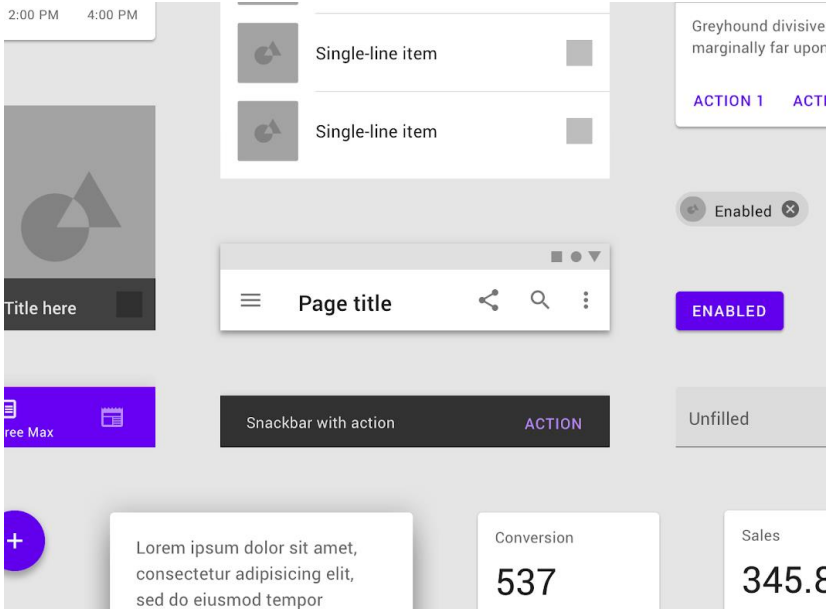
Adaptable system of guidelines, components, and tools that support best practices for UI design

[Material Design homepage]

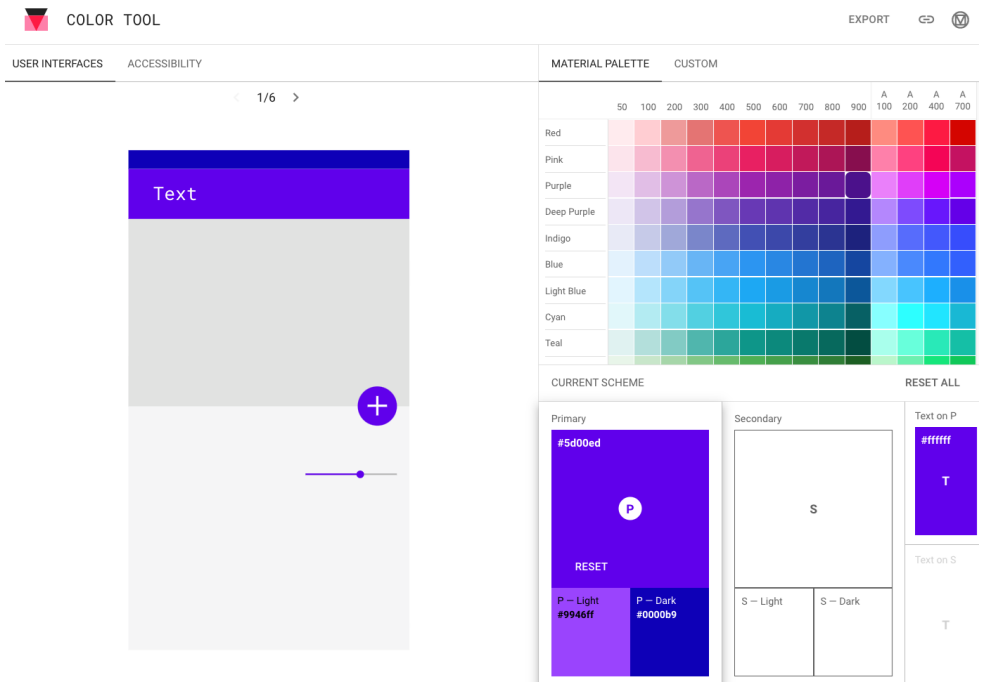


Material Components

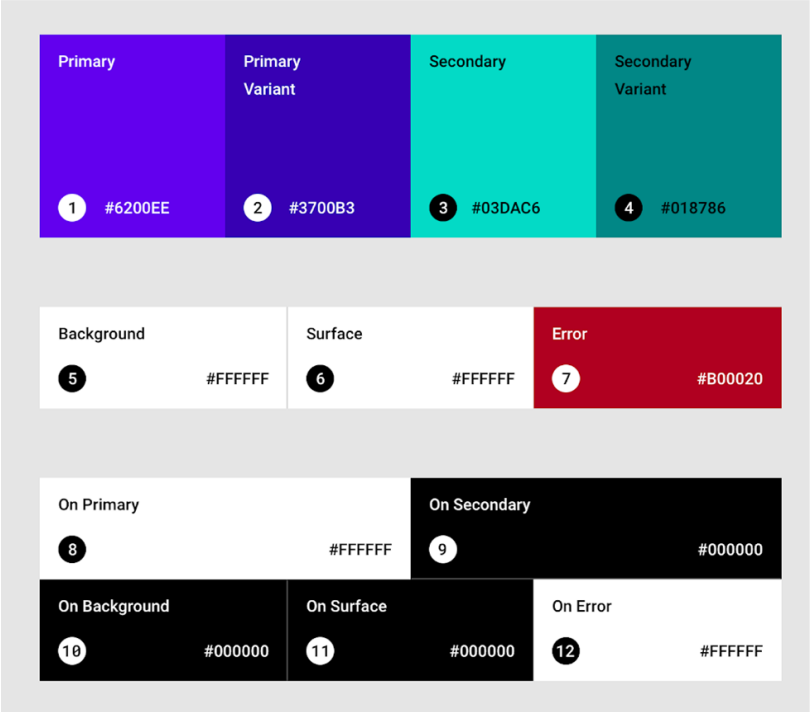
Interactive building blocks for creating a user interface



Material color tool



Baseline Material color theme



Material Components for Android Library

```
implementation 'com.google.android.material:material:<version>'
```

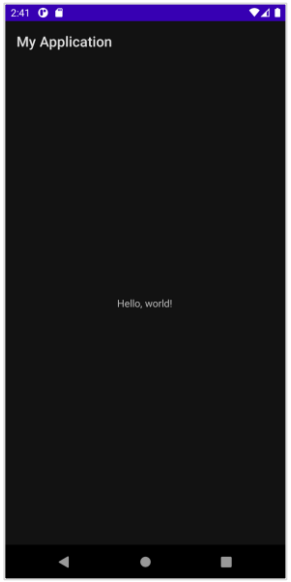
Material Themes

- `Theme.MaterialComponents`
- `Theme.MaterialComponents.NoActionBar`
- `Theme.MaterialComponents.Light`
- `Theme.MaterialComponents.Light.NoActionBar`
- `Theme.MaterialComponents.Light.DarkActionBar`
- `Theme.MaterialComponents.DayNight`
- `Theme.MaterialComponents.DayNight.NoActionBar`
- `Theme.MaterialComponents.DayNight.DarkActionBar`

Material theme example

`Theme.MaterialComponents.DayNight.DarkActionBar`

Light mode



Dark mode

Support dark theme

In `values/themes.xml`:

```
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
    <item name="colorPrimary">@color/orange_500</item>
    ...
```

In `values-night/themes.xml`:

```
<style name="AppTheme" parent="Theme.MaterialComponents.DayNight">
    <item name="colorPrimary">@color/orange_200</item>
    ...
```

Material Components

Material Components

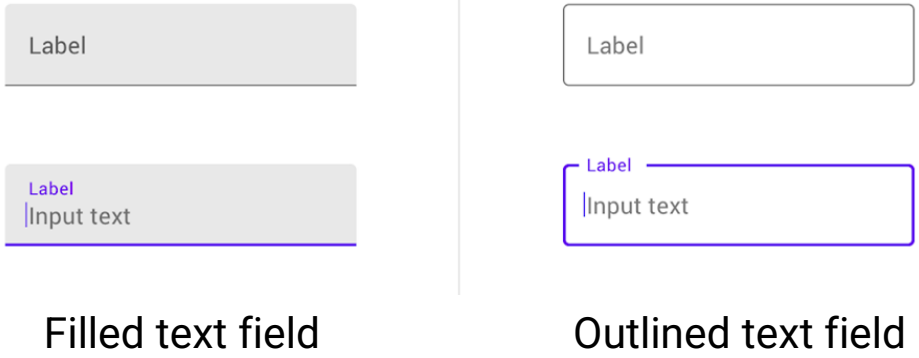
Component library provided for Android and design guidelines

- Text fields
- Buttons
- Menus
- Cards
- Chips
- App bars (top and bottom)
- Floating Action Button (FAB)
- Navigation Drawer
- Bottom navigation
- Snackbar

...and more!

Text field

- Composed of `TextInputLayout` with child view `TextInputEditText`
- Shows a floating label or a text hint before the user enters text
- Two types:



Filled text field

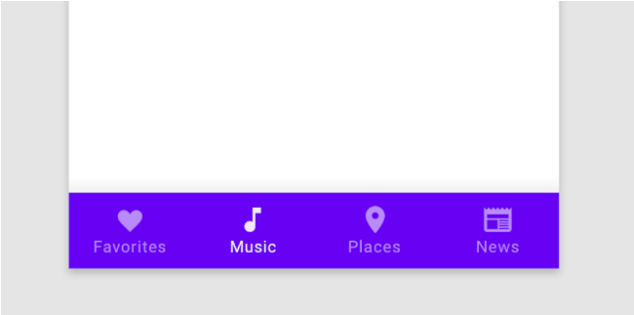
Outlined text field

Text field example

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textField"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/label"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</com.google.android.material.textfield.TextInputLayout>
```

Bottom navigation

- Allows movement between top level destinations in your app
- Alternate design pattern to a navigation drawer
- Limited to 5 locations max



Bottom navigation example

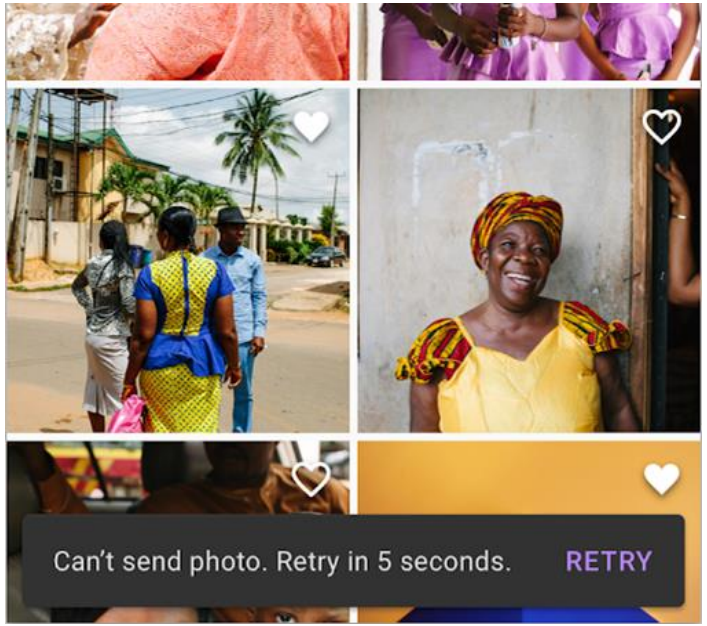
```
<LinearLayout …>

    ...

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:menu="@menu/bottom_navigation_menu" />

</LinearLayout>
```

Bottom navigation listener

```
bottomNav.setOnNavigationItemSelectedListener { item ->
    when(item.itemId) {
        R.id.item1 -> {
            // Respond to navigation item 1 click
            true
        }
        R.id.item2 -> {
            true
        }
        else -> false
    }
}
```

Snackbar

- Display short messages within the app
- Messages have a duration (`SHORT`, `LONG`, or `INDEFINITE`)
- May contain an optional action
- Works best in a `CoordinatorLayout`
- Shown at bottom of enclosing container



Snackbar examples

Show a simple message:

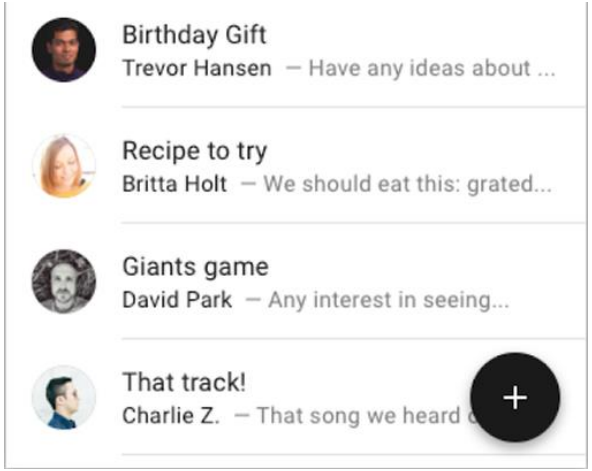
```
Snackbar.make(view, R.string.text_label, Snackbar.LENGTH_SHORT)
    .show()
```

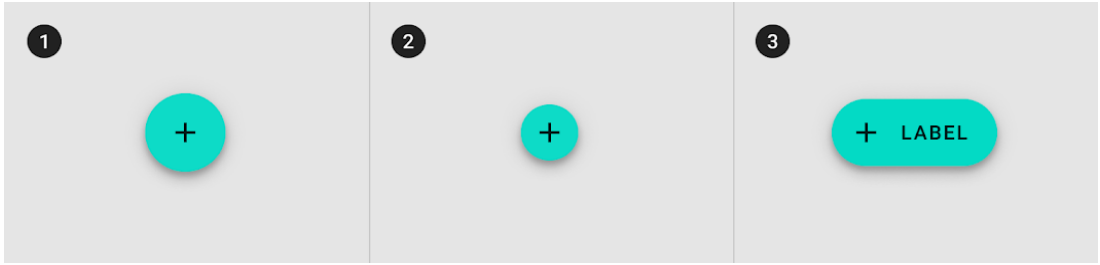
Add an action to a Snackbar:

```
Snackbar.make(view, R.string.text_label, Snackbar.LENGTH_LONG)
    .setAction(R.string.action_text) {
        // Responds to click on the action
    }
    .show()
```

Floating Action Button (FAB)

- Perform the most-common action for a screen (for example, creating a new email)
- Come in multiple sizes (regular, mini, and extended)





CoordinatorLayout

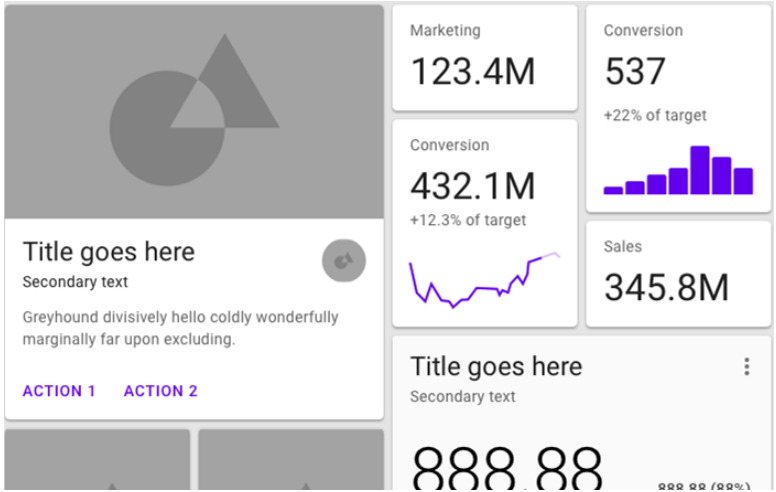
- Acts as top-level container in an app
- Manages interaction of its child views, such as gestures
- Recommended for use with views like a Snackbar or FAB

FAB example

```
<androidx.coordinatorlayout.widget.CoordinatorLayout ...>
    ....
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/floating_action_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:contentDescription="@string/fab_content_desc"
        app:fabSize="normal" <!-- or mini or auto -->
        app:srcCompat="@drawable/ic_plus"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Cards

- A card holds content and actions for a single item.
- Cards are often arranged in a list, grid, or dashboard.
- Use `MaterialCardView`.



MaterialCardView example

```
<com.google.android.material.card.MaterialCardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView .../>
        <TextView .../>
    </LinearLayout>

</com.google.android.material.card.MaterialCardView>
```

Note about fragments

Use the AndroidX version of the `Fragment` class. (`androidx.fragment.app.Fragment`).

Don't use the platform version of the `Fragment` class (`android.app.Fragment`), which was deprecated.