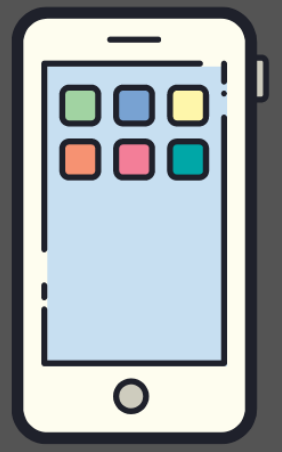


MOBILE DEVELOPMENT

Basic Layout Development 1

Outline

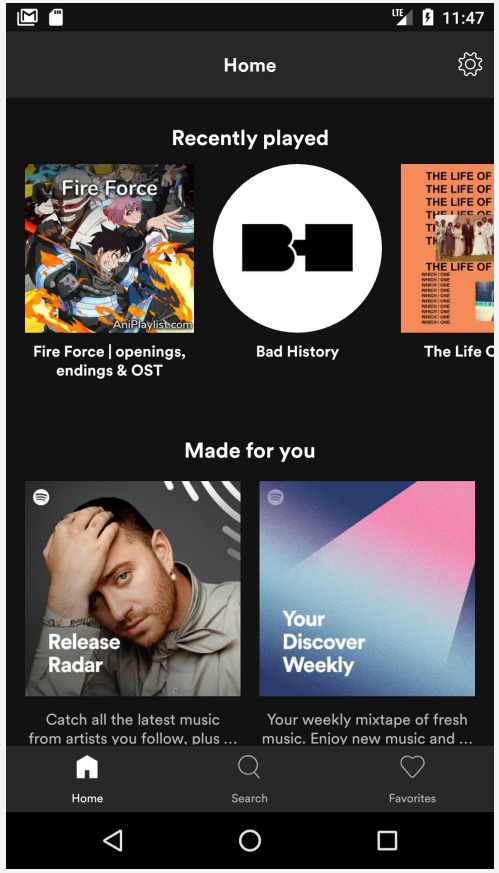
- Android Basics: User Interface
  - Resources
  - Views (UI elements)
  - ViewGroups / Layouts

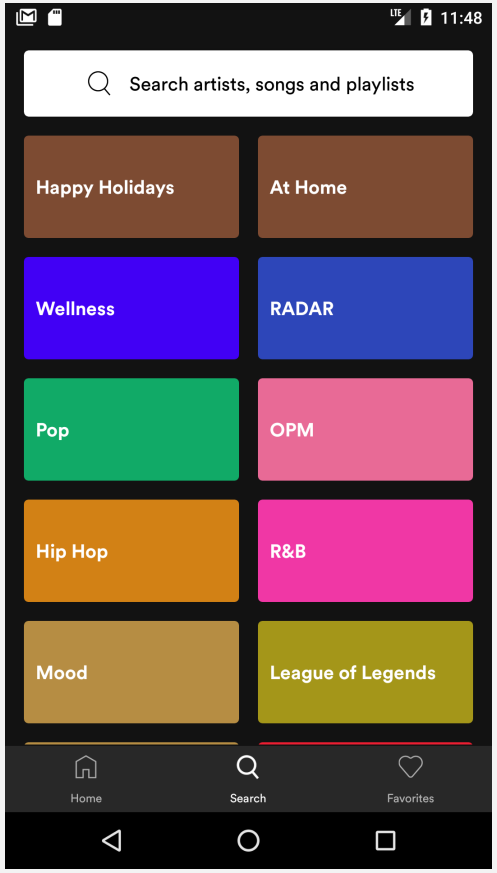
Take a moment to think:

What app(s) do you commonly use?

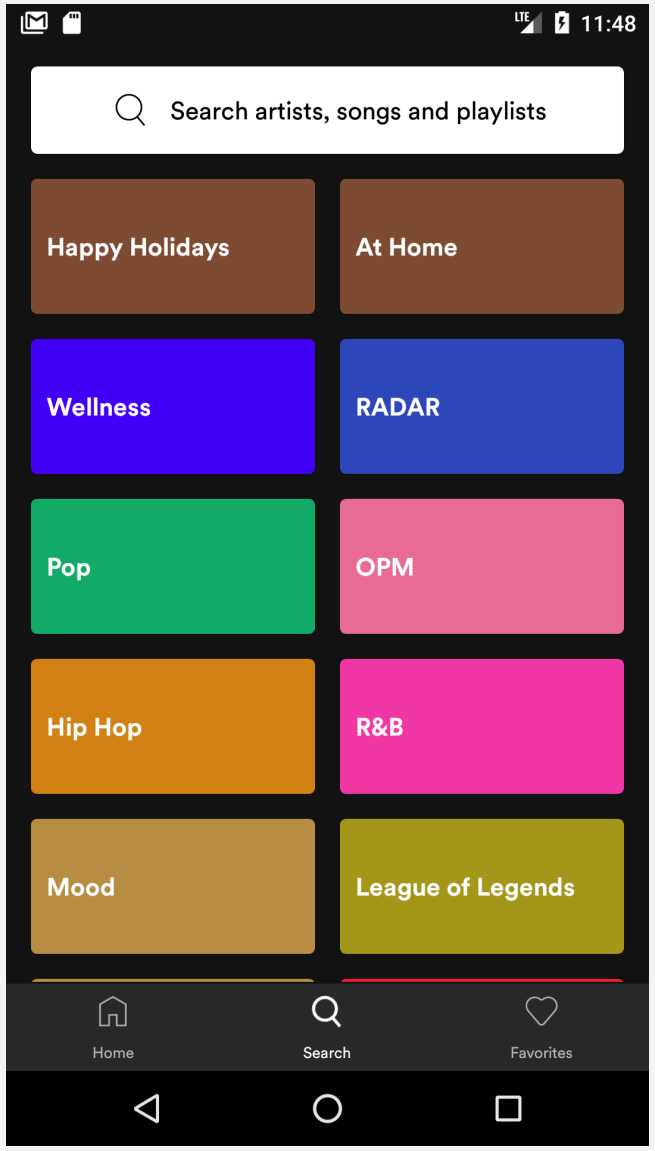
Motivation: Think of their UI

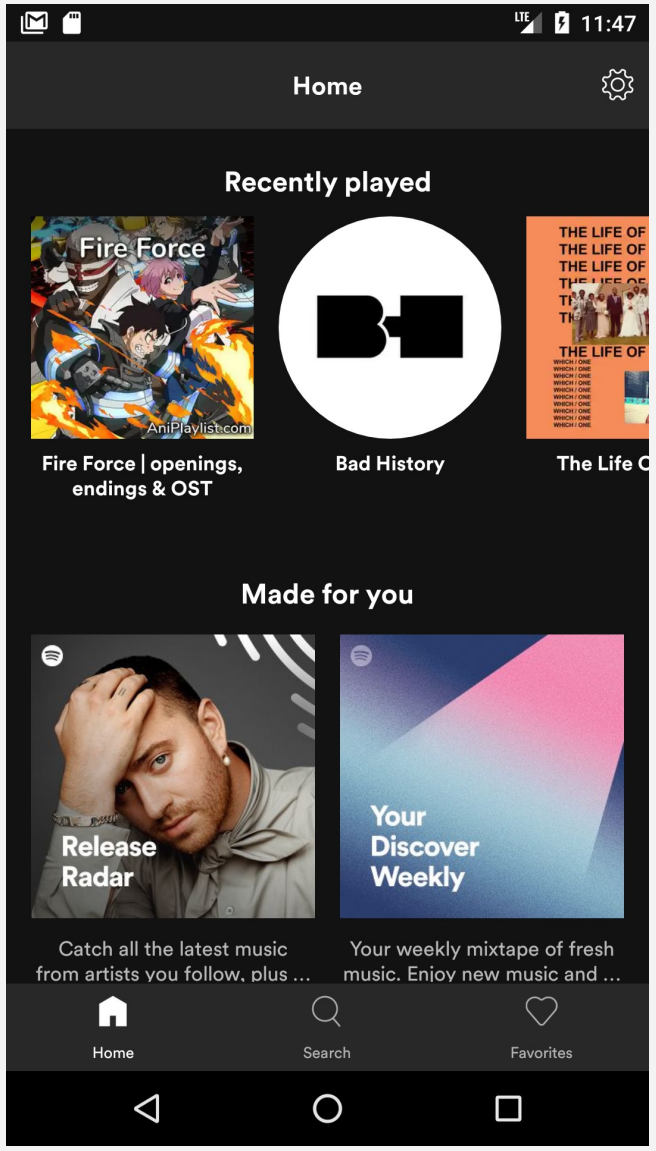
- We can learn a lot the user-facing aspect just by looking at our commonly used apps



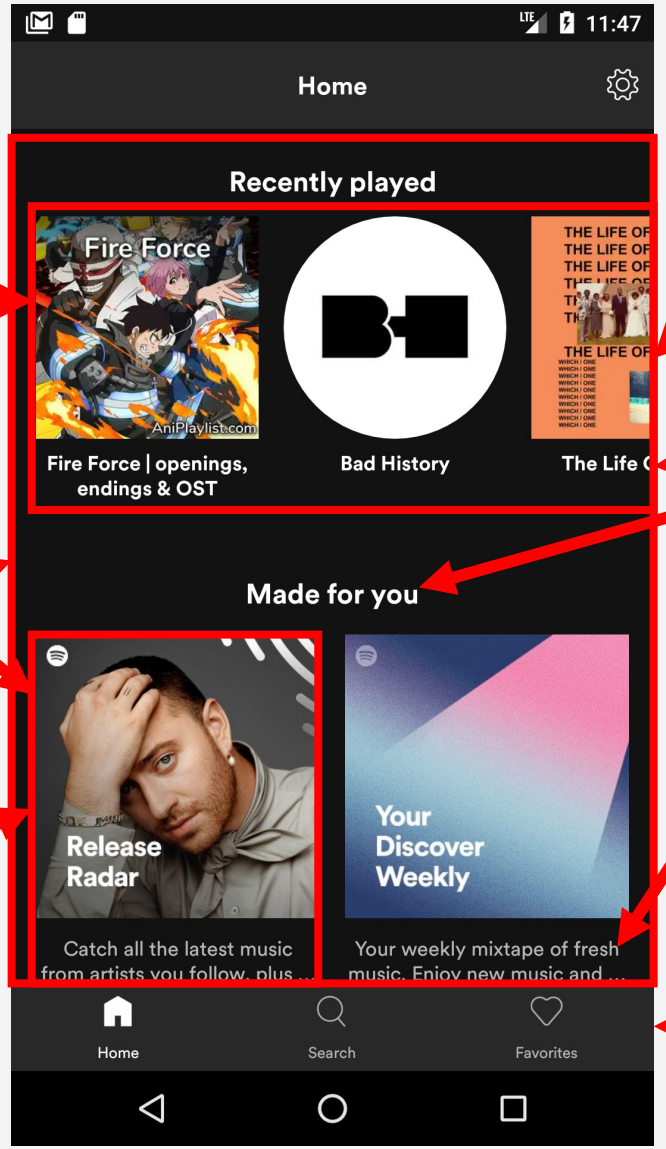


What UI elements can you identify?





What UI elements can you identify?



Image

Scrolling area (vertical)

Button-like (i.e. clickable)

List / scrolling area (horizontal)

Text / Label

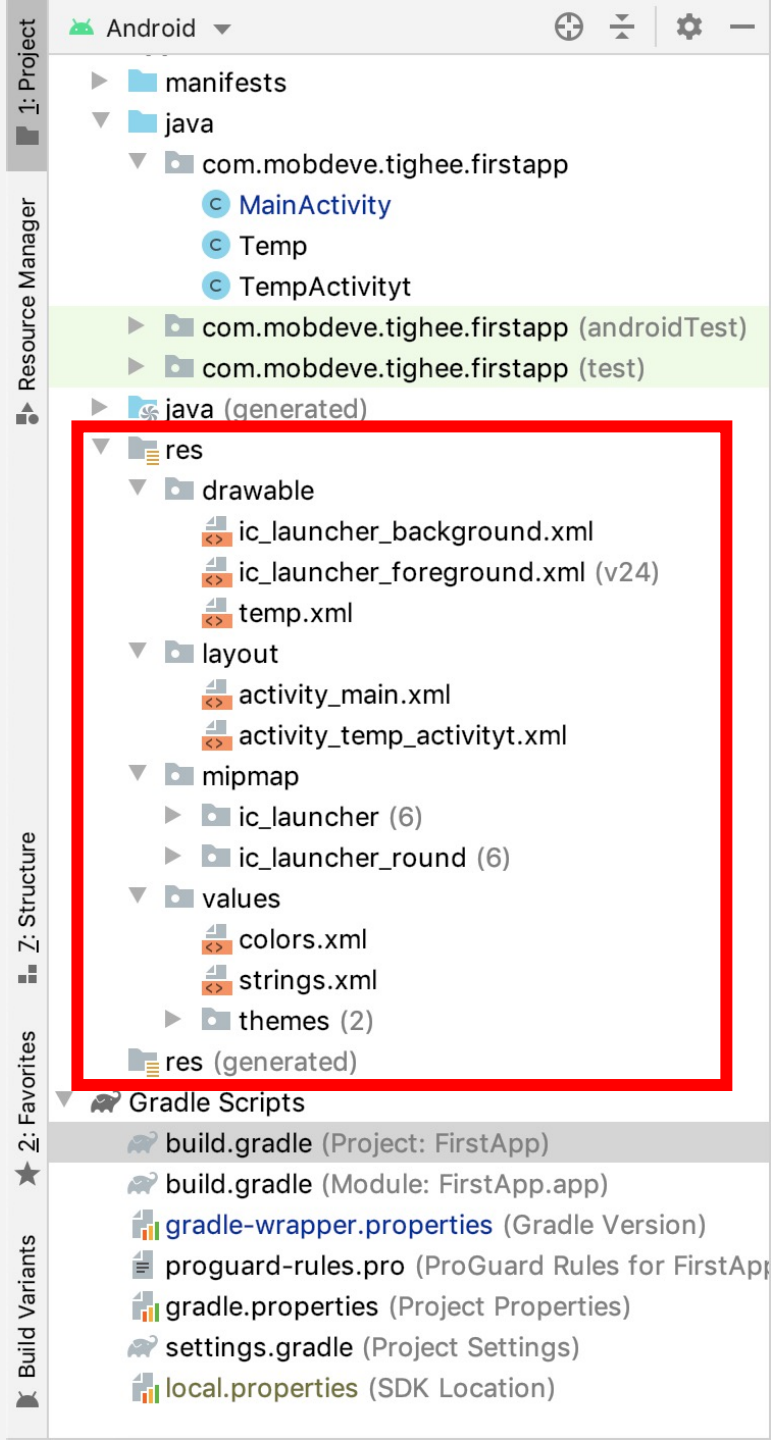
Navigation

Motivation

- What are UI elements with respect to Android?
- How do we create UI elements?

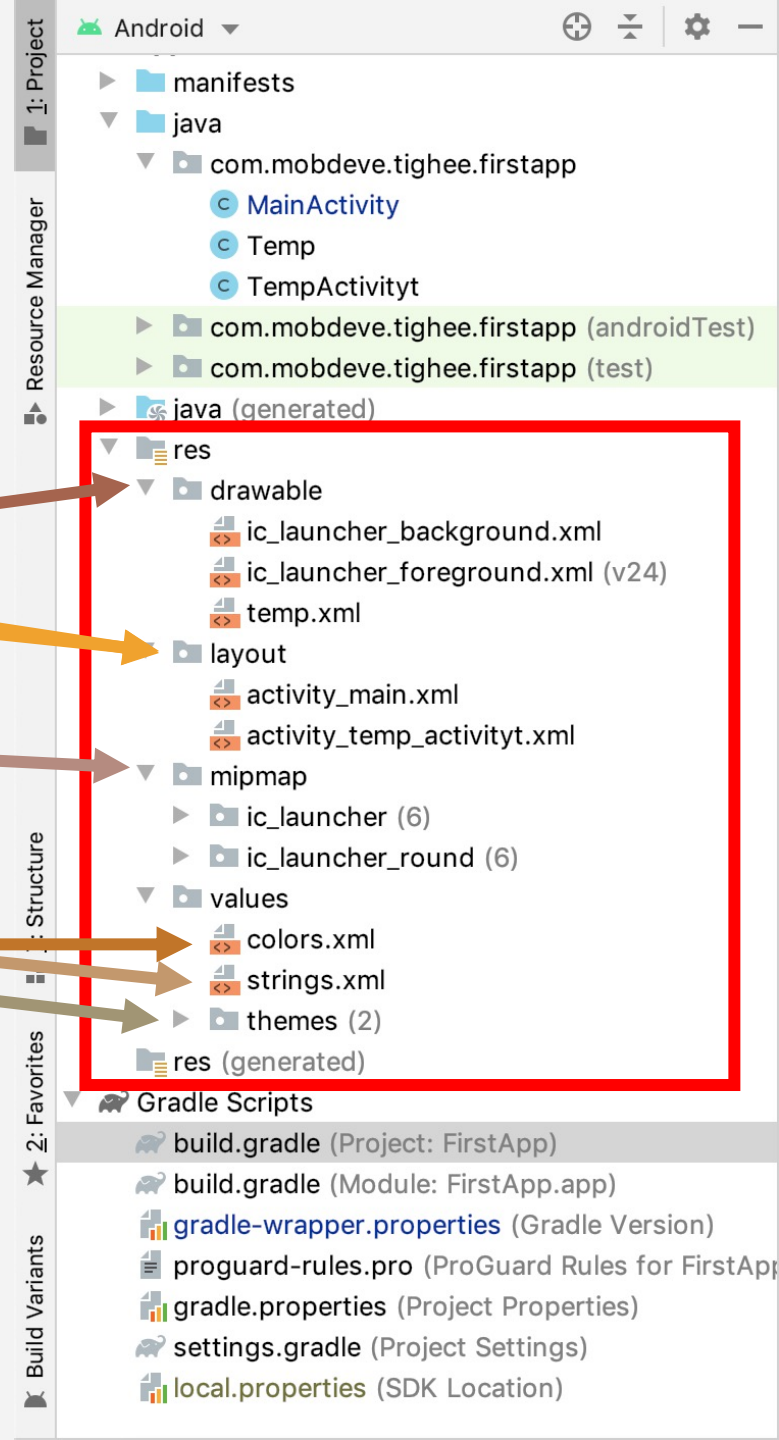
What is a resource?

- Elements that are external to or separate from the app's behavior (i.e. code)
- Consist of
  - Images
  - XML files
- Recall: Found in the res folder

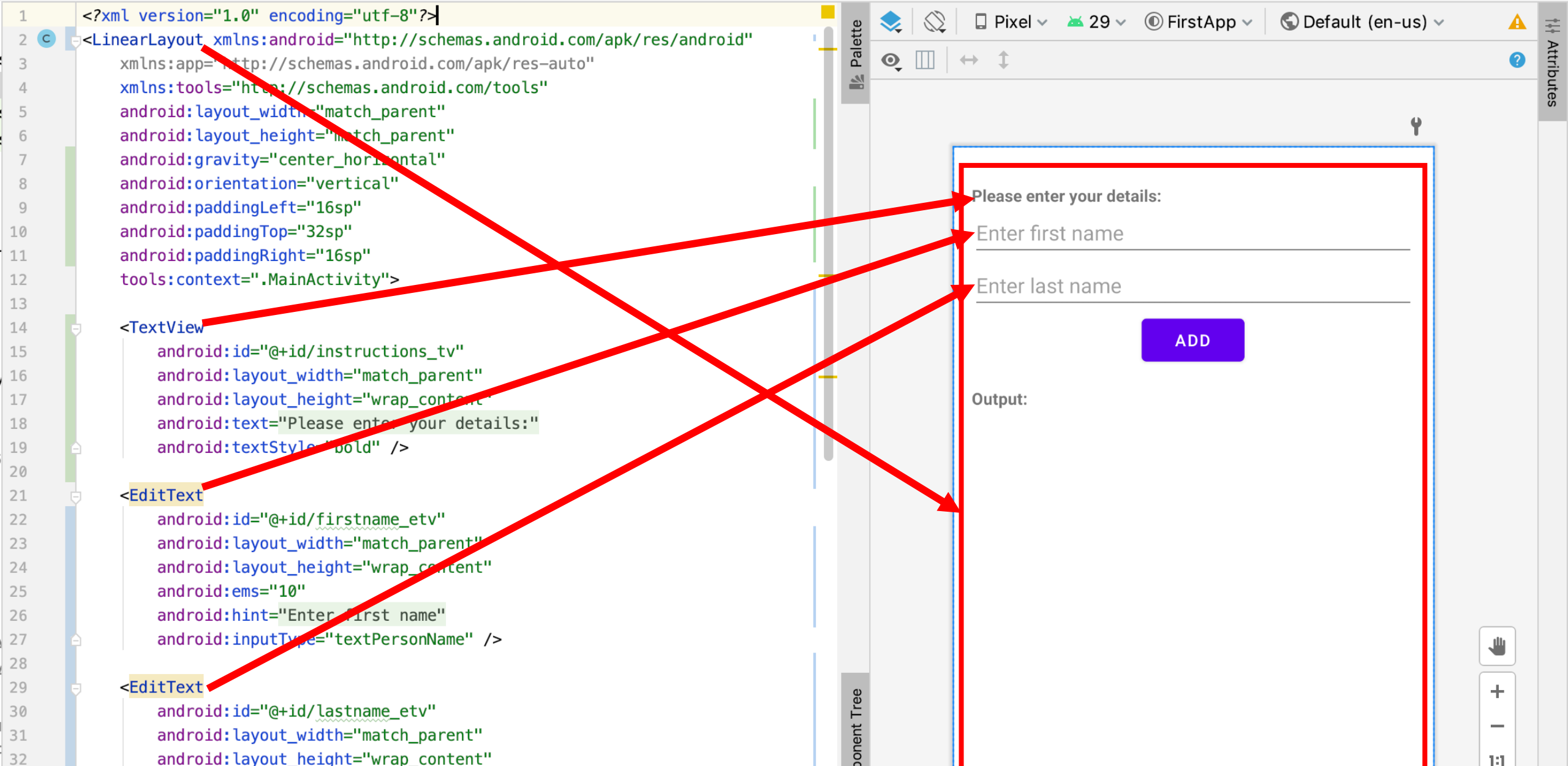


What should be a resource?

- Layouts
- Controls (buttons, etc.)
- Images
- Strings
- Style and theming information



UI components are written in XML files



Views

- View is the superclass for visual interface elements
  - I.e. Elements you see onscreen
- Common Views include:
  - TextViews
  - EditTexts [input]
  - ImageViews
  - Buttons
- All behave the same, but subclasses have “extra” features

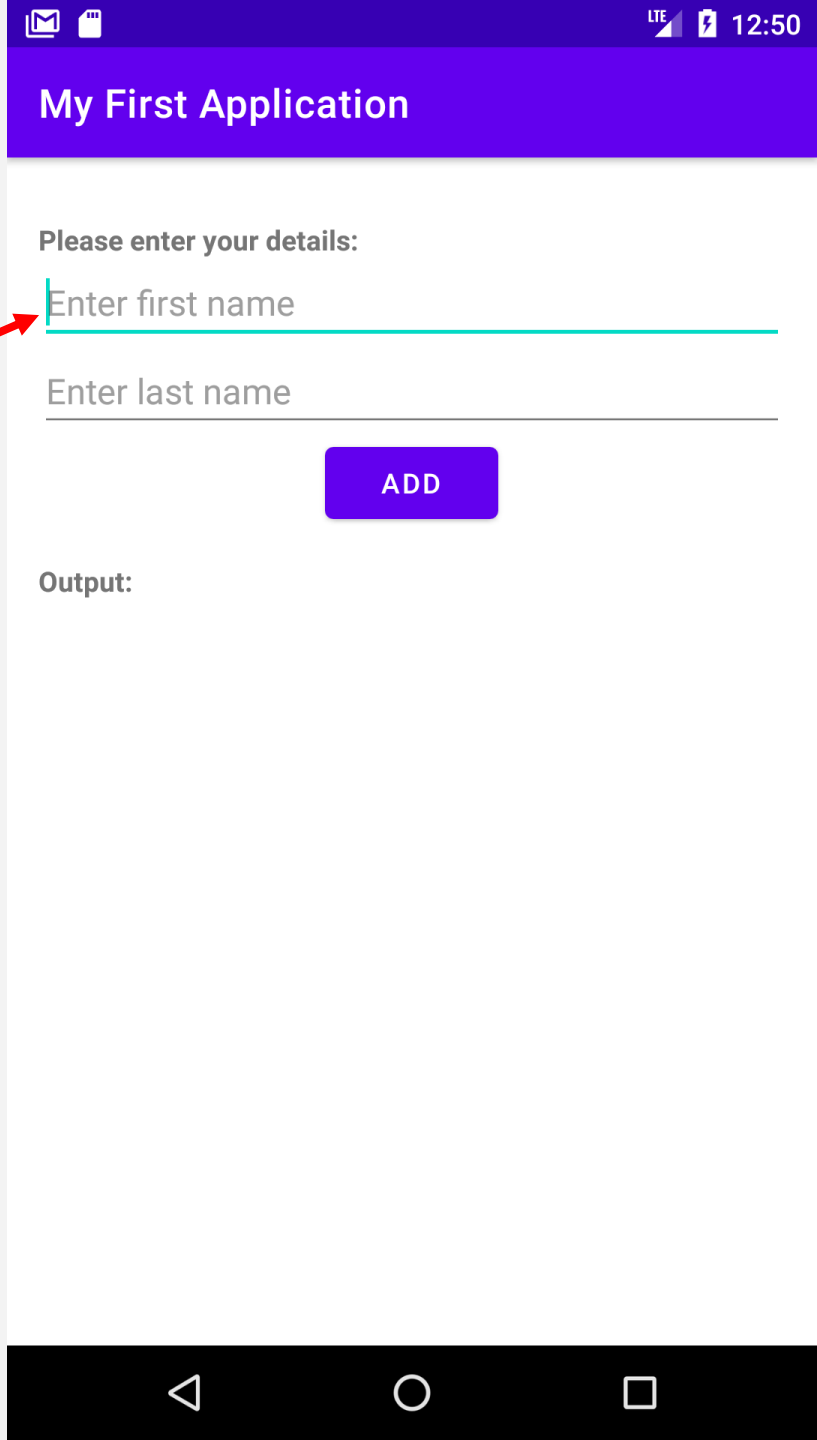
These are commonly called Widgets

But these are different from app widgets!

Views

- All Views have…
  - Width and height
  - Padding
  - OnClick [listener] Not just buttons!
- As for subclasses with “extra” features…
  - EditTexts allow for hints
  - ImageViews don’t have text

Hints disappear unlike the text attribute



```
<?xml version="1.0" encoding="utf-8"?>
<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:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingLeft="16sp"
    android:paddingTop="32sp"
    android:paddingRight="16sp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/instructions_tv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/instructions_tv"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/firstname_etv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/first_edit_text"
        android:inputType="textPersonName" />
```

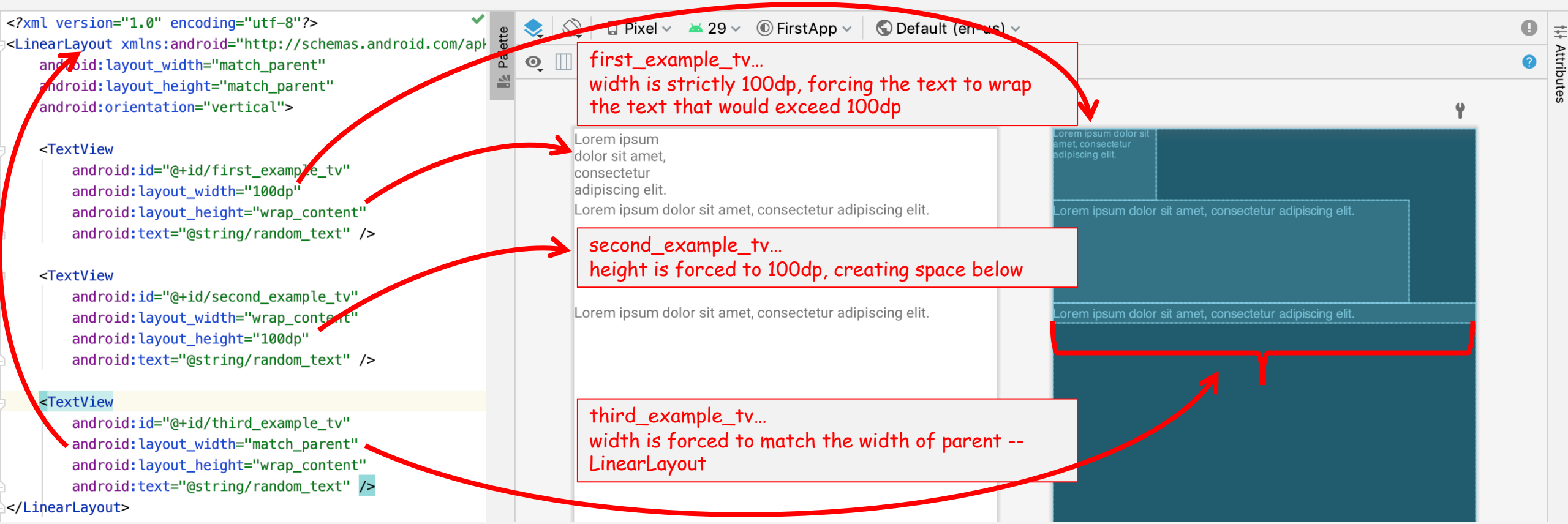
View Properties

- Just like any object, Views have attributes / properties
- Common attributes you'll frequently modify:
  - id notice all ids start with **@+id/**
  - layout_width
  - layout_height

View Properties – Height / Width

- Both layout_height and layout_width are used to specify the size of a View with respect to a ViewGroup / layout
- Typically:
  - A specific value (e.g. 12dp)
  - wrap_content – wrap dimension based on content of View
  - match_parent – dimension matches that of the parent View

View Properties – Height / Width



View Properties – Units

Q: Which unit should normally be used?

A:
sp and **dp** are good. sp usually for text, dp used for when you don't want the views be to be resized
px and **pt** are allowed, but are discouraged

- dp is a "density-independent pixel"
  - On a 160-dpi (dots-per-inch) screen, 1 dp equals 1px (pixel)
  - As dpi increases, the number of pixels per dp increases
- px is an actual screen pixel
- sp is a "scale-independent pixel"
  - Like dp but scaled by the system's font preference
  - 1sp will cover more dp in a device set at larger font
- pt is 1/72 of an inch of the physical screen

How about images?

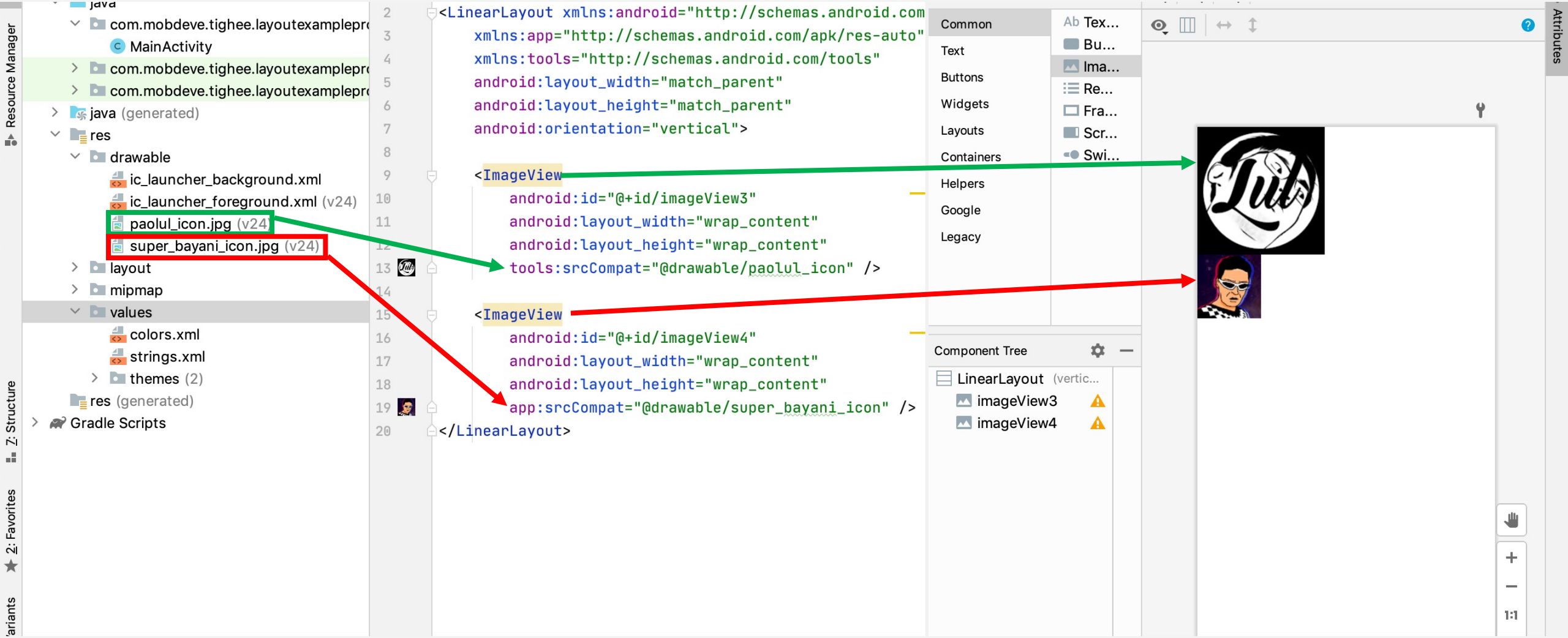
Before hand…

- Let's understand that there are two types of image assets:
  - Drawables
    - Contains graphics (PNG, JPEG, etc.)
    - Can be created through File > New > Image Asset or by dragging/saving images into the project's drawable folder
  - Mipmap
    - Launcher icon files in different resolutions
    - Can be created through File > New > Image Asset

There are also multiple drawable / mipmap folders that map image assets to a phone's resolution

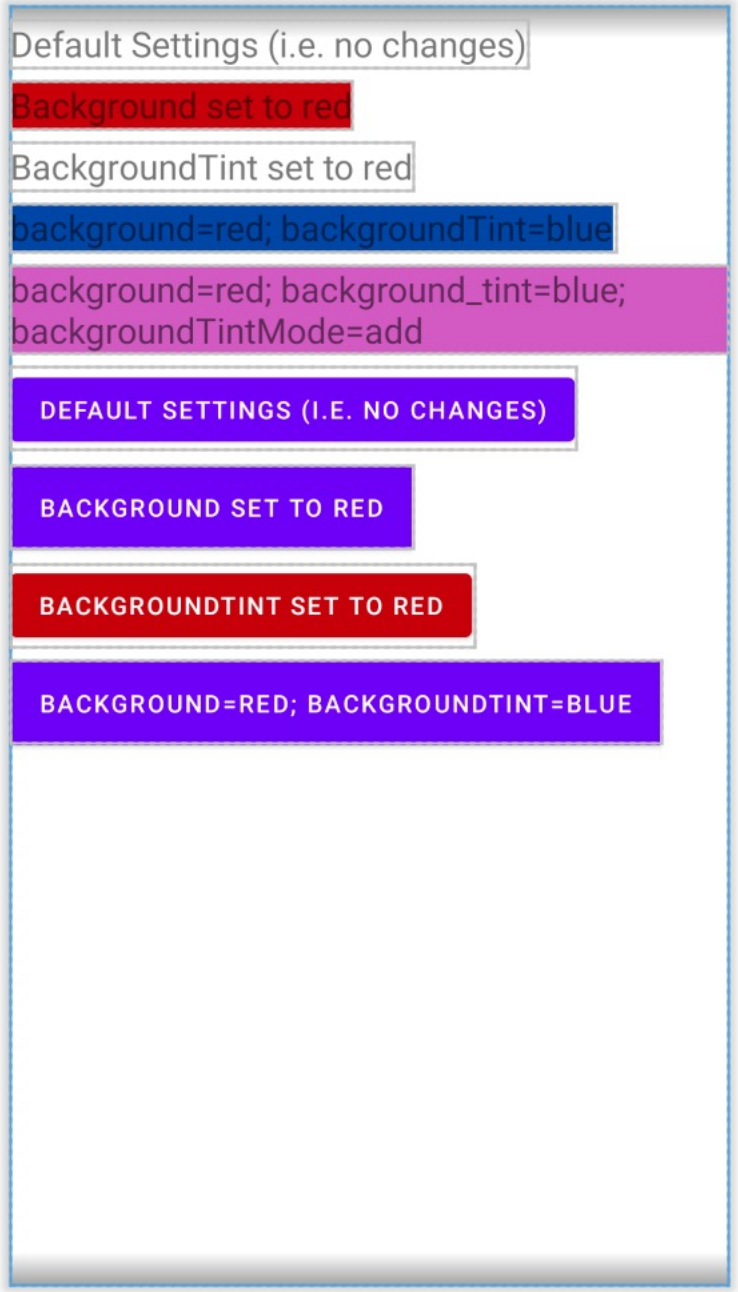
If an image is too high-res for a device, an error will be thrown and the app will crash ☹

Adding Image Files to ImageViews



View Properties – Background + Tint

- When it comes to coloring backgrounds, things might not seem so straight forward
- The ~~two~~three attributes you'd want to play around with are:
  - background -> refers to a drawable or color
  - backgroundTint -> tint applied to the background
    - backgroundTintMode -> blending applied to the tint
      - Add, src_in, src_over, screen, src_atop, multiply



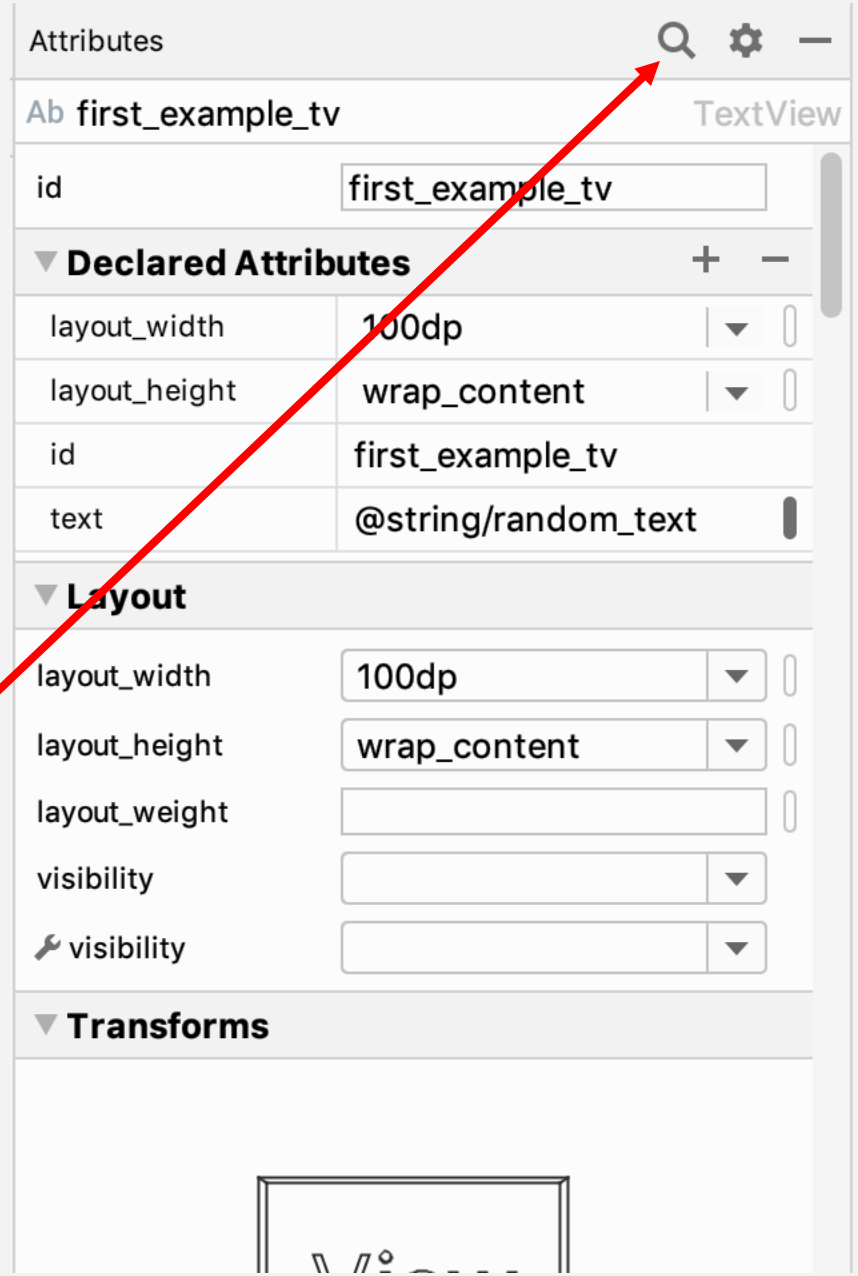
Observe the layout to the left…

- For TextViews (at most other Views):
  - Background sets a color
  - BackgroundTint only has an effect when there's a background
    - By default, there is no background; hence, why the 3rd TextView has no color, but the 4th is changed to blue
  - BackgroundTintMode blends background and tint
- For Buttons:
  - By default, uses the app's colorPrimary
  - Background, by default, is set to @empty and changing it removes the feathered style
  - BackgroundTint changes the color

View Properties

- There are many more properties you'll interact with, like…
  - Margins, Padding
  - Text, Text size, Text style
  - Visibility, Enabled
- Take some time to read the documentation or use the search bar to look for attributes you think might exist

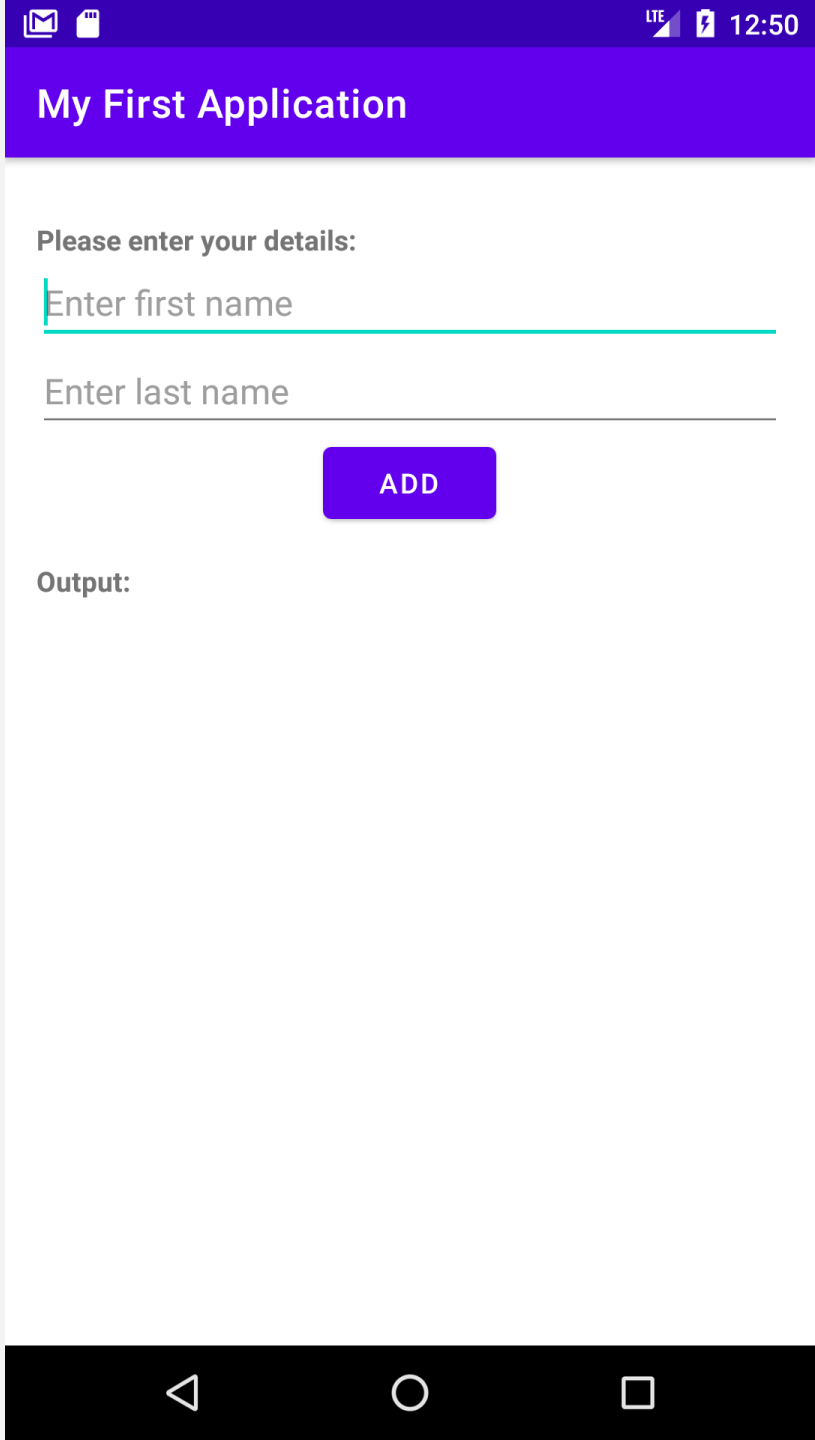
Experimentation is key!



Any questions so far?

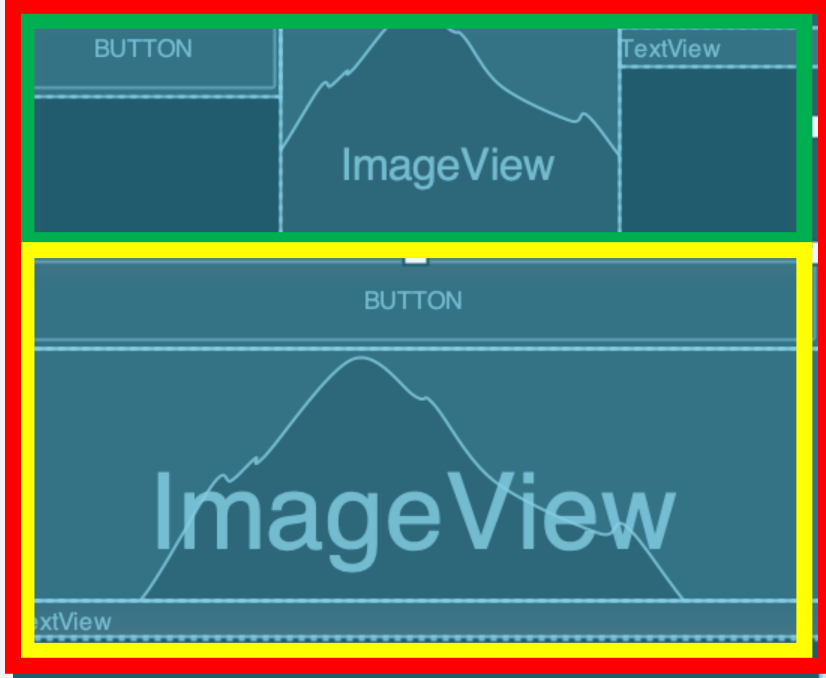
ViewGroups (or Layouts)

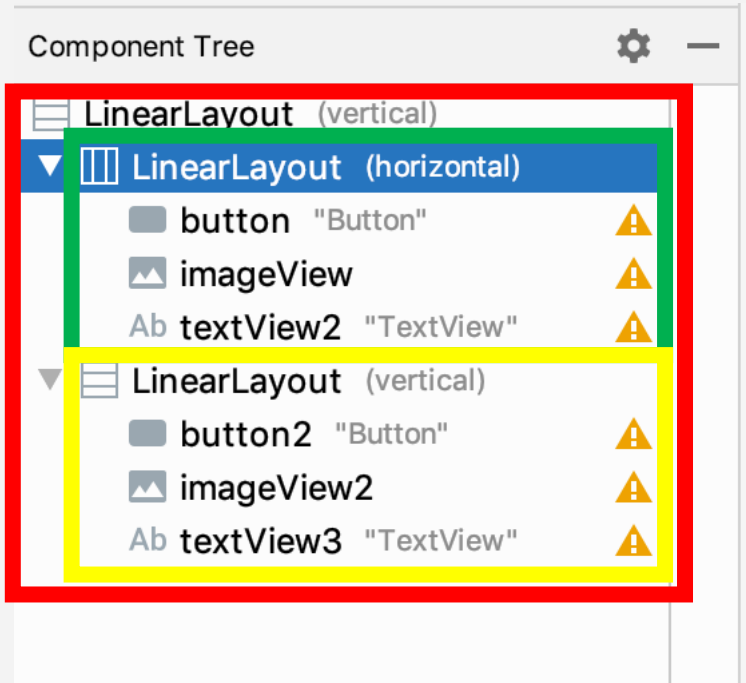
- A ViewGroup is a View that can contain other “child” Views
  - It can also contain other ViewGroups since its also a View
- Common ViewGroups include:
  - LinearLayout [horizonal / vertical]
  - ConstraintLayout



ViewGroups (or Layouts)

- LinearLayouts order Views in a line / single direction
  - orientation specifies the direction
    - horizontal
    - vertical





To have proper spacing, you'd want to play around with layout_weight, margin, padding of either the parent or child views

ViewGroups (or Layouts)

- ConstraintLayout sets relationships between Views using constraints or relative positioning

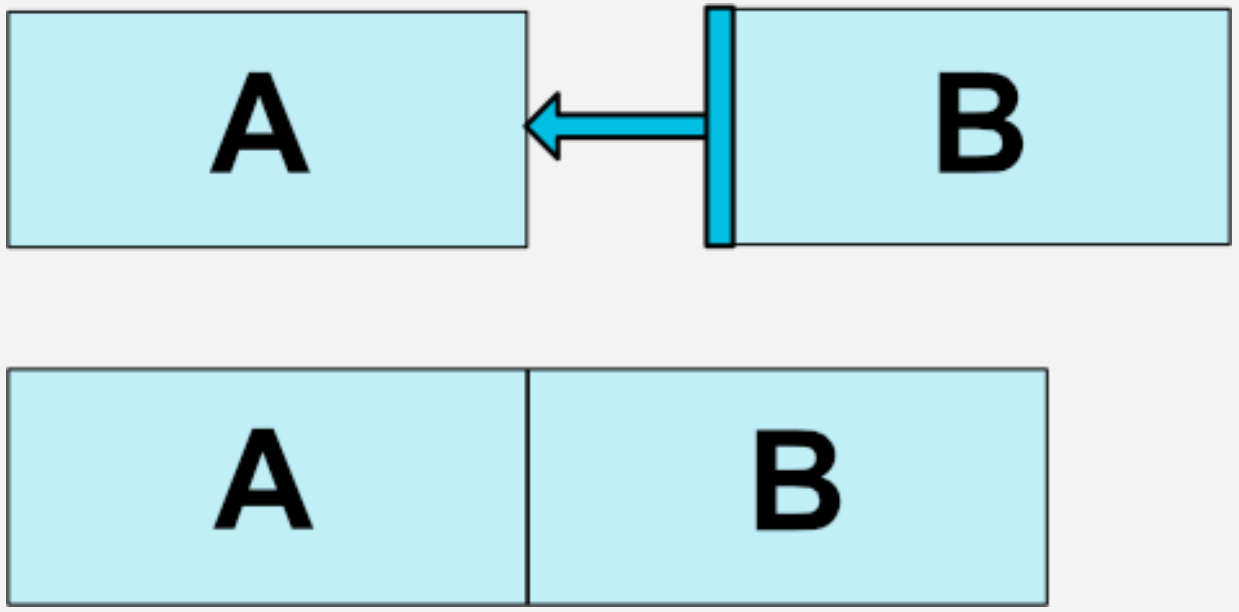
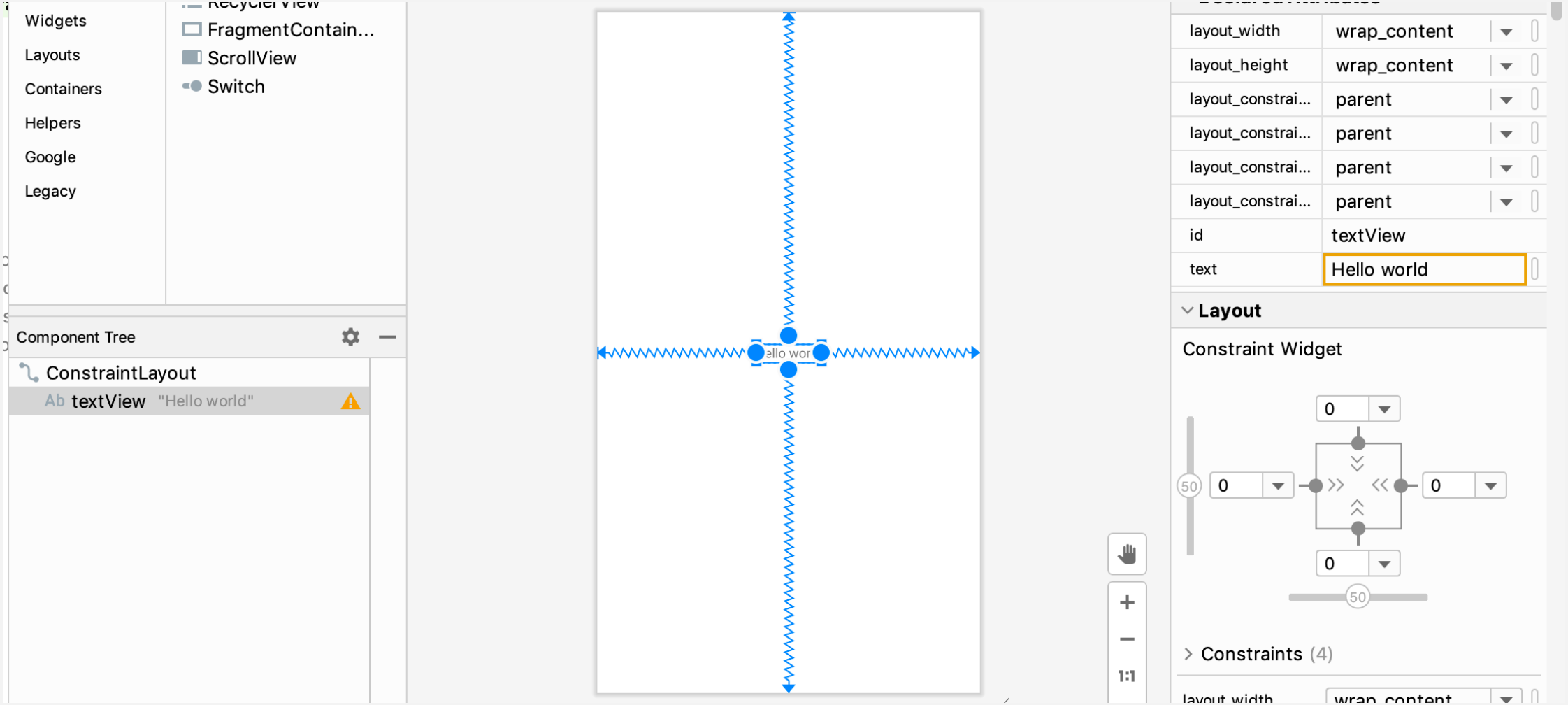


Image from https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout?hl=hi

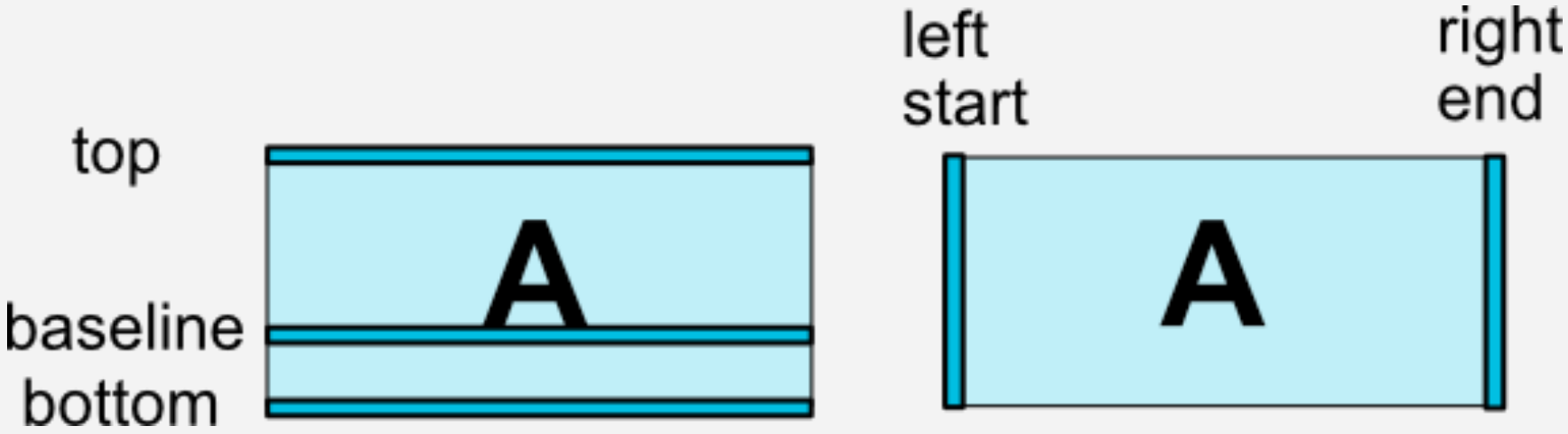
In this example, we see a TextView constrained to its parent from all sides



By "pulling" it with equal force from both sides, the TextView is centered both vertically and horizontally

When we talk about “sides”…

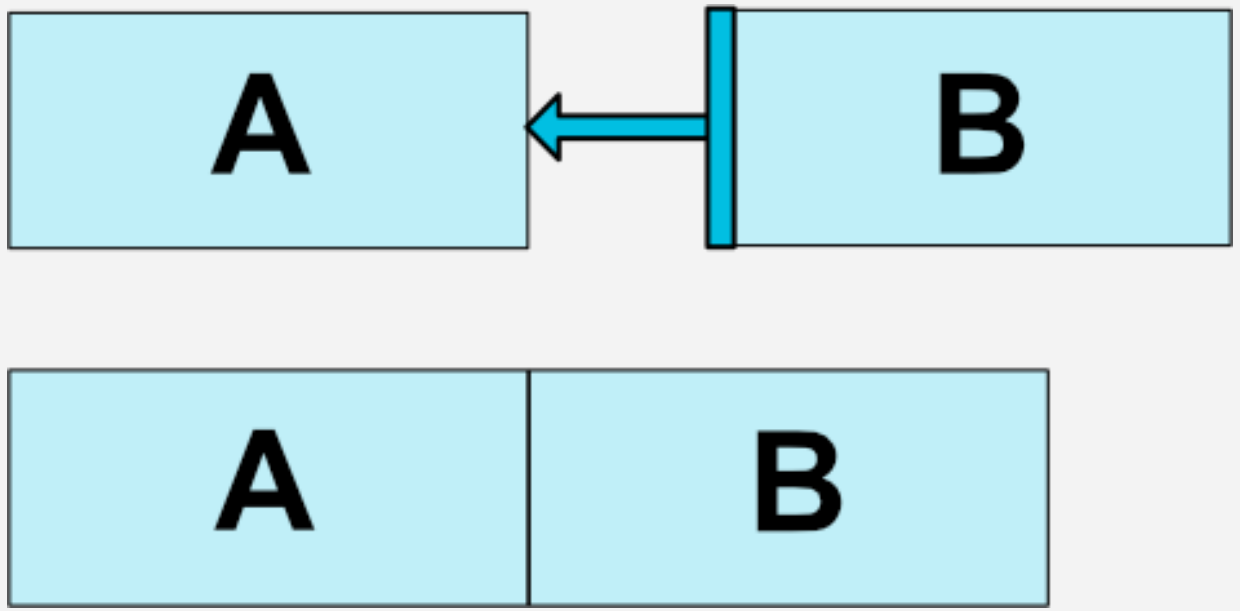
- We actually refer to different positions of a view



start and end are preferred over left and right, as this supports RTL (right to left) layouts

Image from https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout?hl=hi

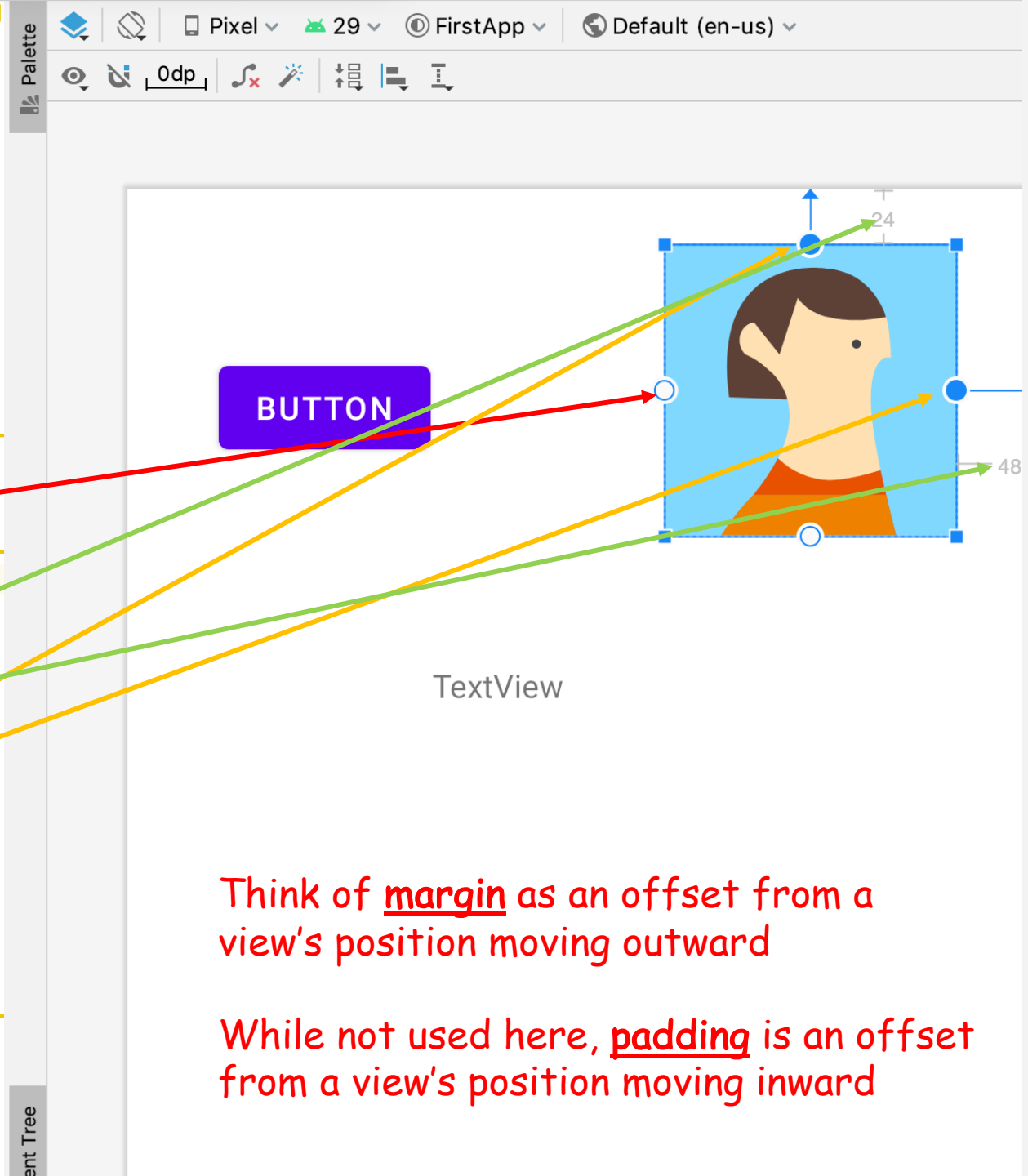
Q: So in our previous example, how is B constrained to A?



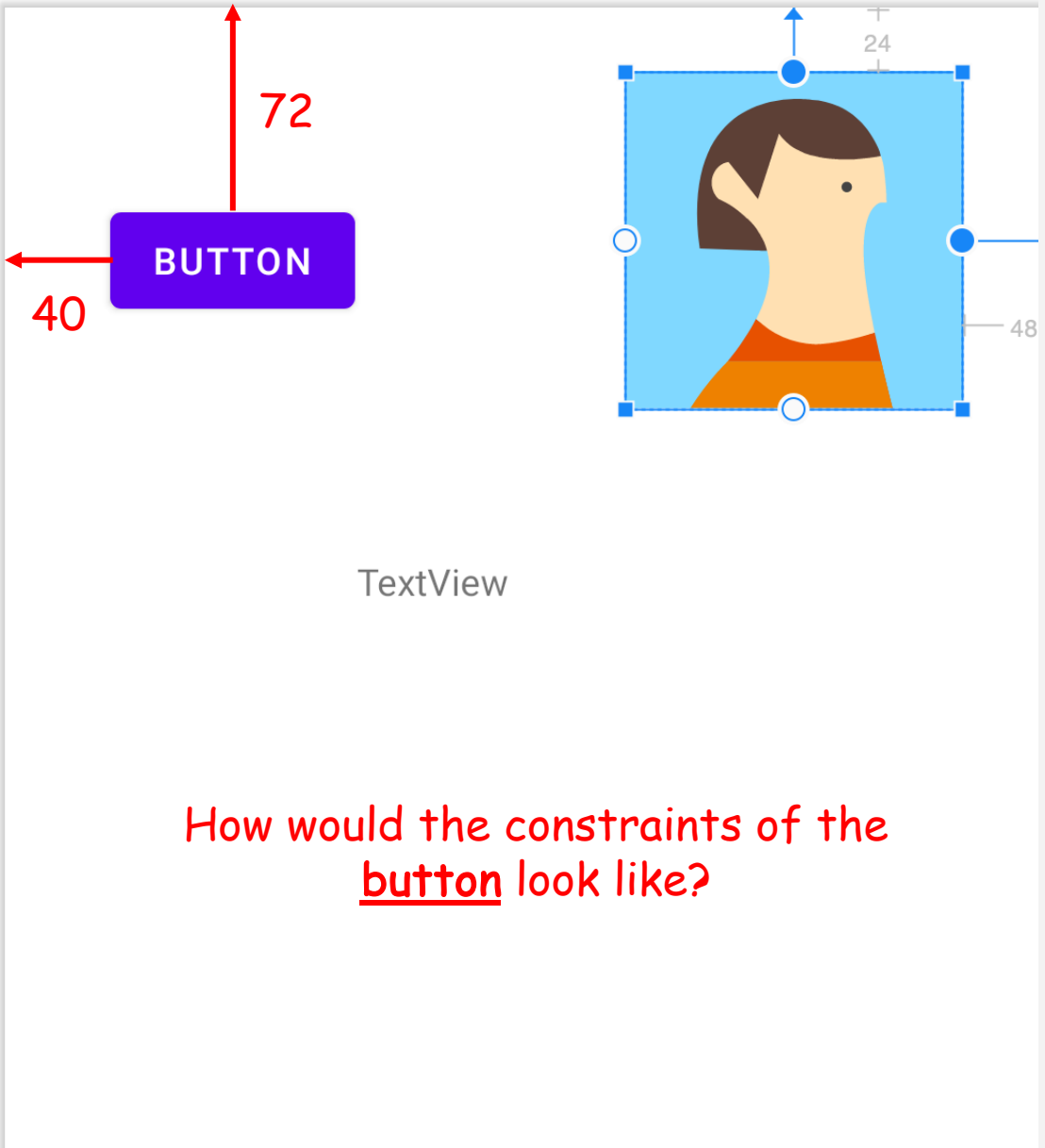
A: B's start is constrained to A's end

Image from https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout?hl=hi

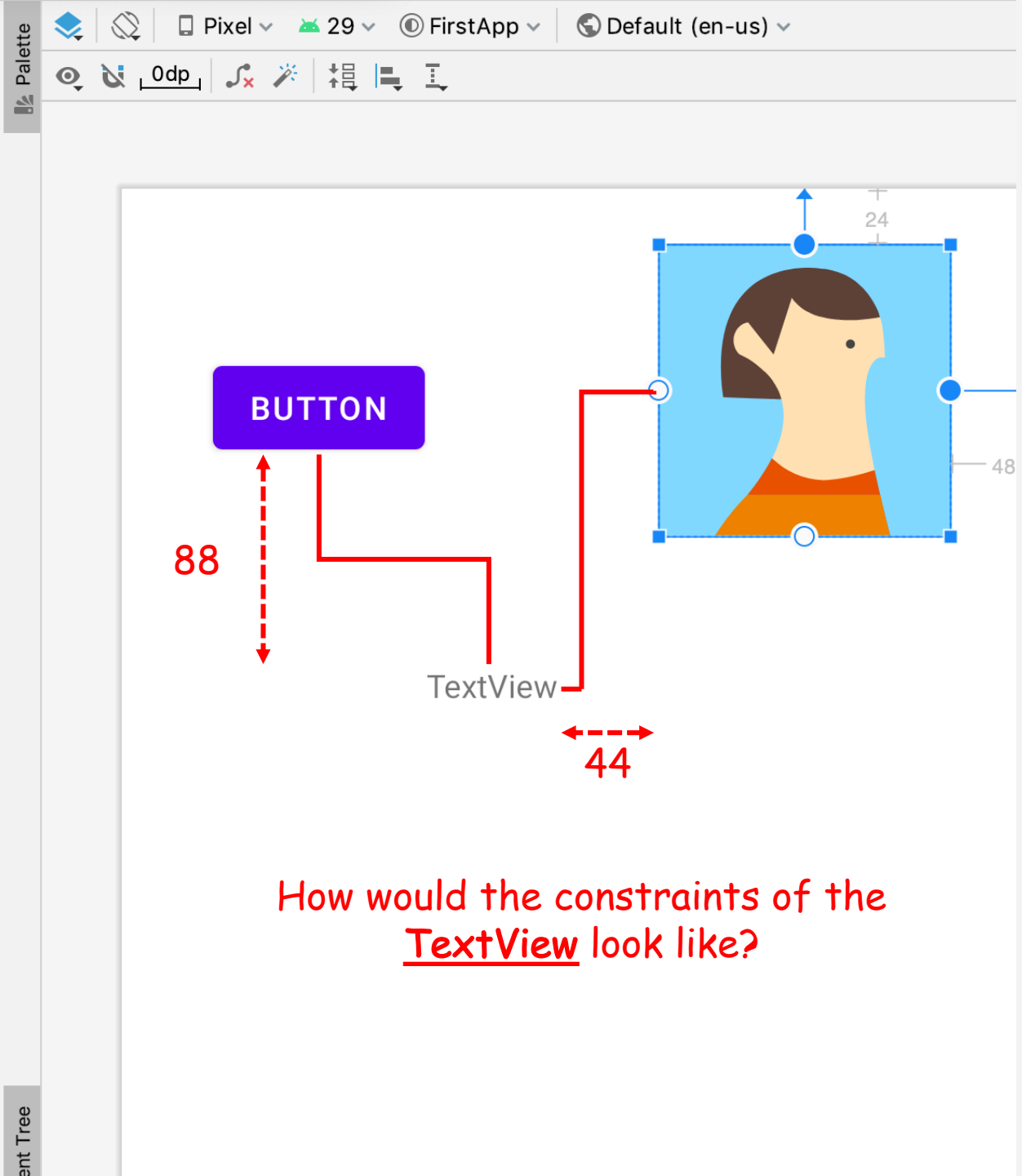
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="40dp"
        android:layout_marginTop="72dp"
        android:text="Button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imageView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="88dp"
        android:layout_marginEnd="44dp"
        android:text="TextView"
        app:layout_constraintEnd_toStartOf="@+id/imageView4"
        app:layout_constraintTop_toBottomOf="@+id/button4" />
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="40dp"
        android:layout_marginTop="72dp"
        android:text="Button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imageView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="88dp"
        android:layout_marginEnd="44dp"
        android:text="TextView"
        app:layout_constraintEnd_toStartOf="@+id/imageView4"
        app:layout_constraintTop_toBottomOf="@+id/button4" />
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="40dp"
        android:layout_marginTop="72dp"
        android:text="Button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imageView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="88dp"
        android:layout_marginEnd="44dp"
        android:text="TextView"
        app:layout_constraintEnd_toStartOf="@+id/imageView4"
        app:layout_constraintTop_toBottomOf="@+id/button4" />
```



[Modified] View Properties – Height / Width

- Both layout_height and layout_width are used to specify the size of a View with respect to a ViewGroup / layout
- Typically:
  - A specific value (e.g. 12dp)
  - wrap_content – wrap dimension based on content of View
  - match_parent – dimension matches that of the parent View
    - Discouraged when using a ConstraintLayout
  - 0dp (match_constraints) – takes all available space based on constraints

There’s actually a lot more to discover when it comes to styling with ConstraintLayouts

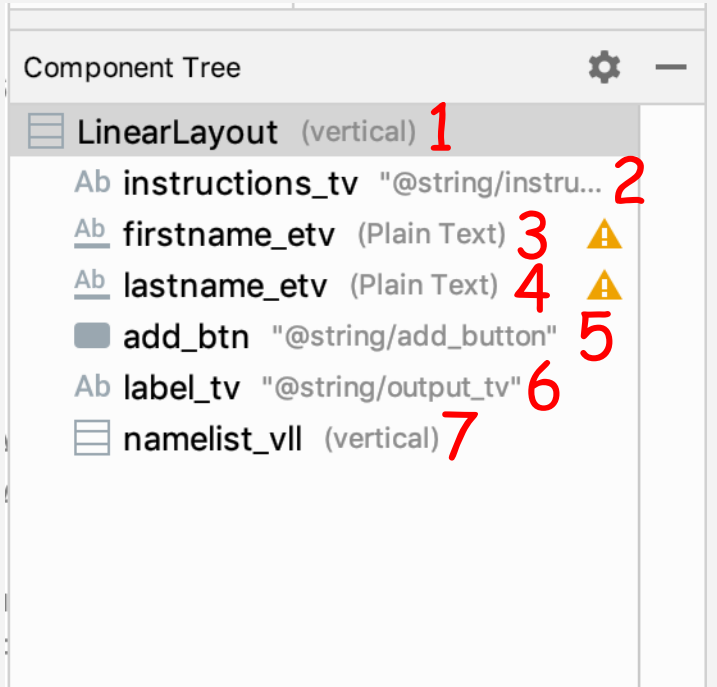
For more info, check out

https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout

ViewGroups (or Layouts)

- Q: Can you guess the number of layouts use in the screenshot to the side?
  - There are actually two LinearLayouts!
- The arrangement corresponds to a tree structure known as the Component tree

In a ConstraintLayout, order doesn't matter



Any questions? ☺

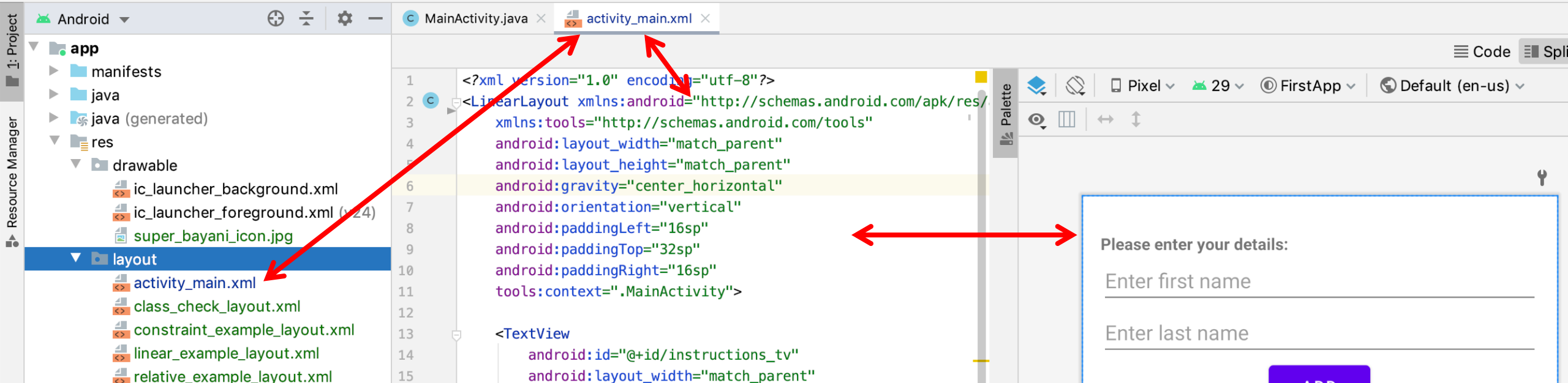
Non-graded Exercise

- Try to recreate the layout file shown on the right side
- There are different ways to solve this, so experiment with different ideas based on what you've learned!
- There is also a Canvas assignment for this

Icon from Super Bayani, Subscribe @ https://www.youtube.com/channel/UCGn0ep0f0tjnegstKIlVgqg

Clarification: Layouts

- A Layout file is an XML file composed of Views and ViewGroups (Layout)
  - Has a root ViewGroup
  - Usually associated to an Activity but not required to be

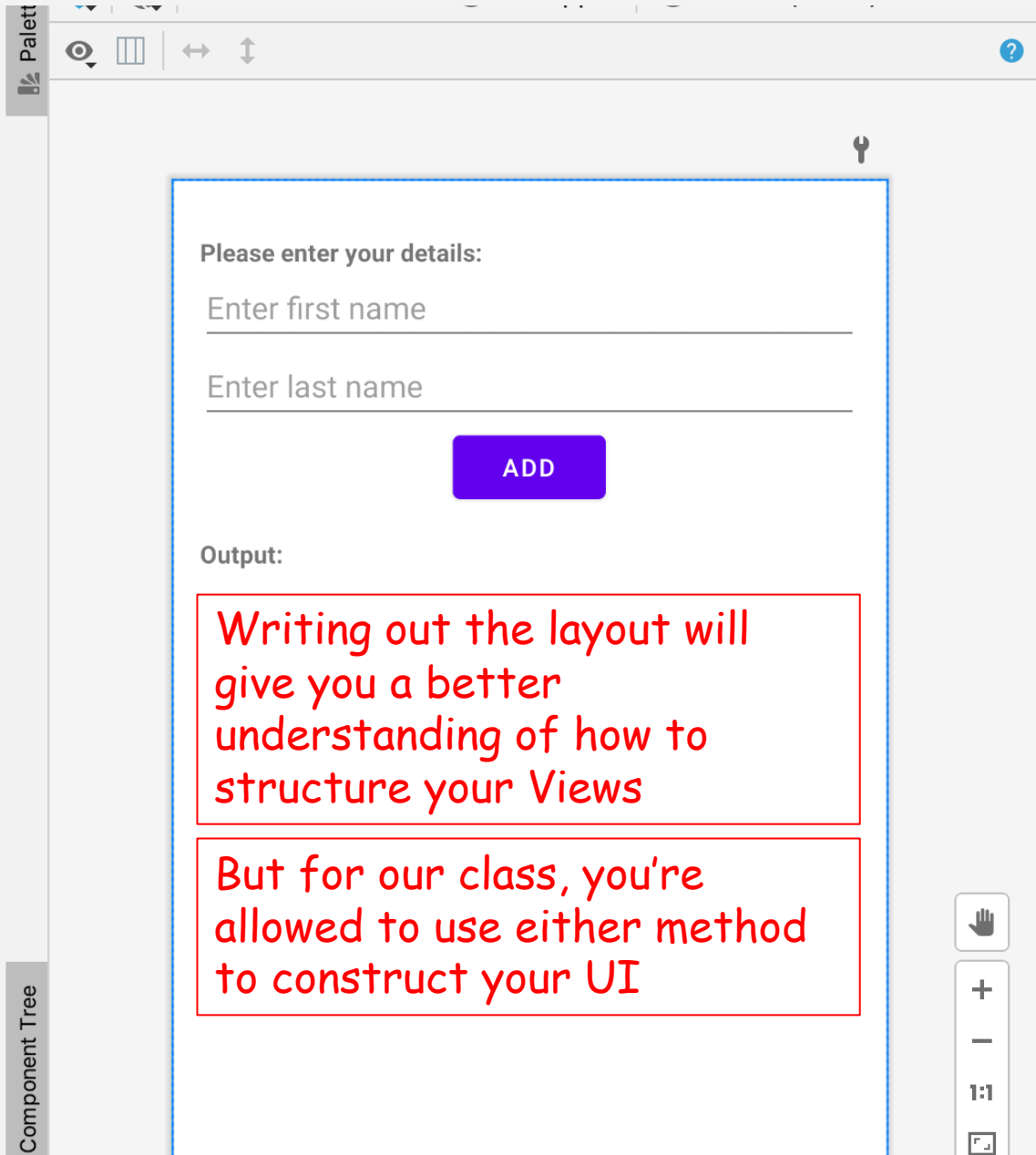


Clarification: Creating UI / Layouts

```
<LinearLayout 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"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingLeft="16sp"
    android:paddingTop="32sp"
    android:paddingRight="16sp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/instructions_tv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Please enter your details:"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/firstname_etv"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter first name"
        android:inputType="textPersonName" />

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

Its common to just use the **Design View** to generate XML, but be careful of what you select

The attribute *ems* actually doesn't need to be here because we're using *match_parent* as the width

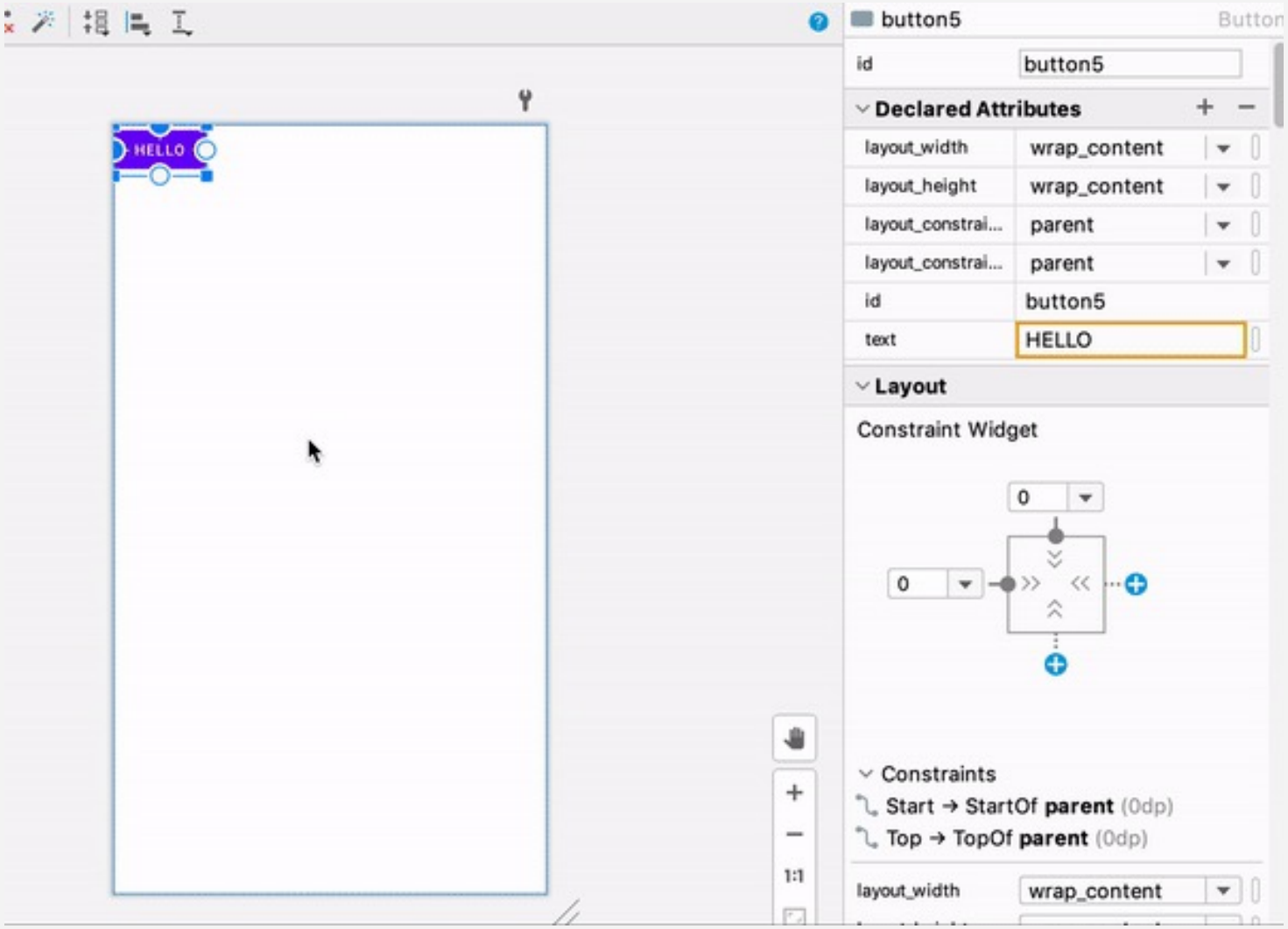


Writing out the layout will give you a better understanding of how to structure your Views

But for our class, you're allowed to use either method to construct your UI

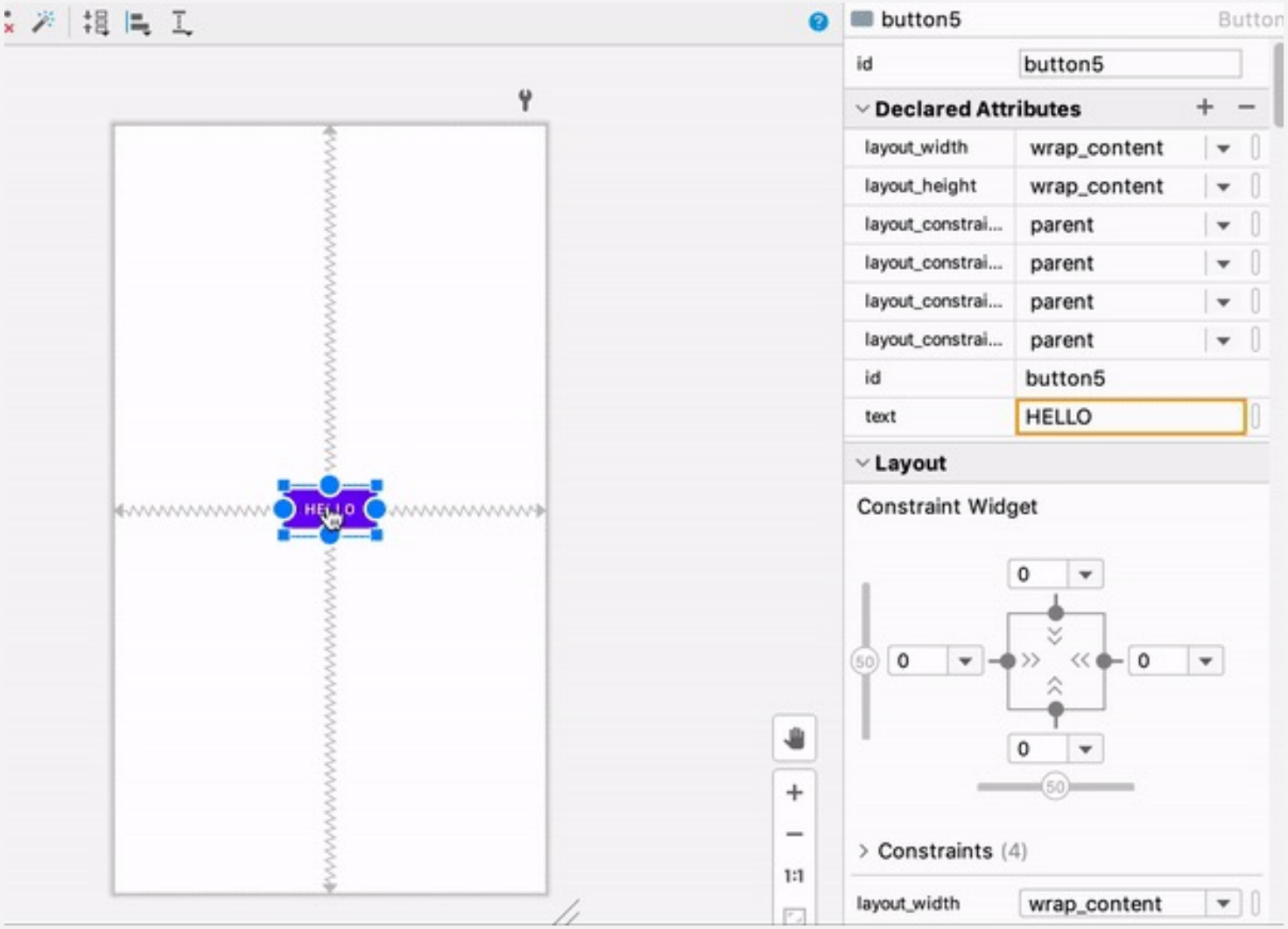
Clarification: Moving Views in a ConstraintLayout

- In the example to the side, we see moving the button in design mode makes changes to the margin



Clarification: Moving Views in a ConstraintLayout

- In this next example to the side, we see moving the fully constrained button in design mode makes changes to the bias



Any questions? ☺

Notes

- There is an Android Project in Module 2a where you can access all the layouts shown in this slide
- We'll release exercise #1 after next session
  - Look to clarify concerns next meeting
- For the next lesson, we'll tackle the 2nd part of UI Basics: Dynamic creation of views

Thanks everyone!

See you next meeting! ☺

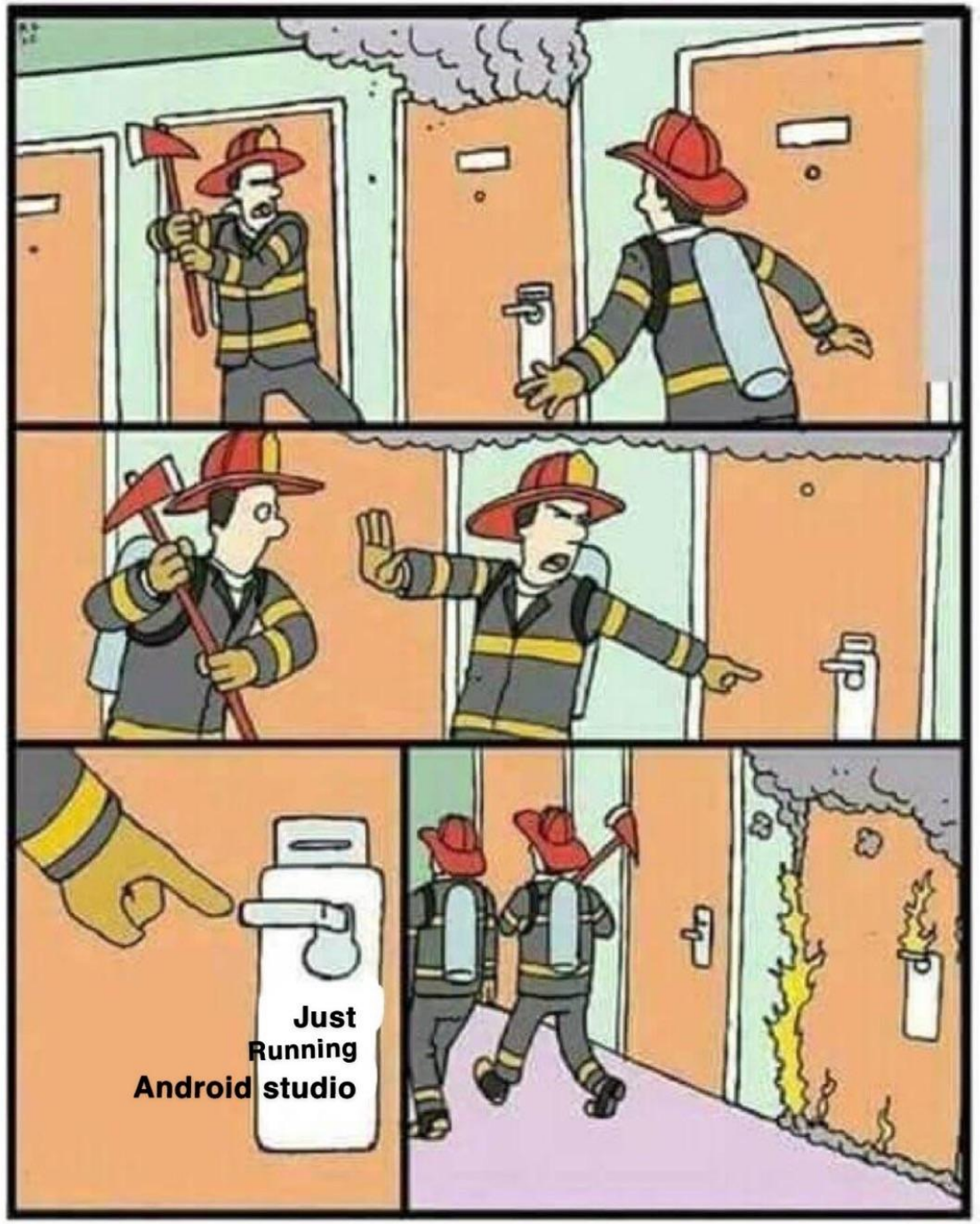


Image sauce: https://www.reddit.com/r/ProgrammerHumor/comments/afhbqz/gotta_love_android_studio/