CMPS 312

Read Chapter 1 & 6

Views & Layout

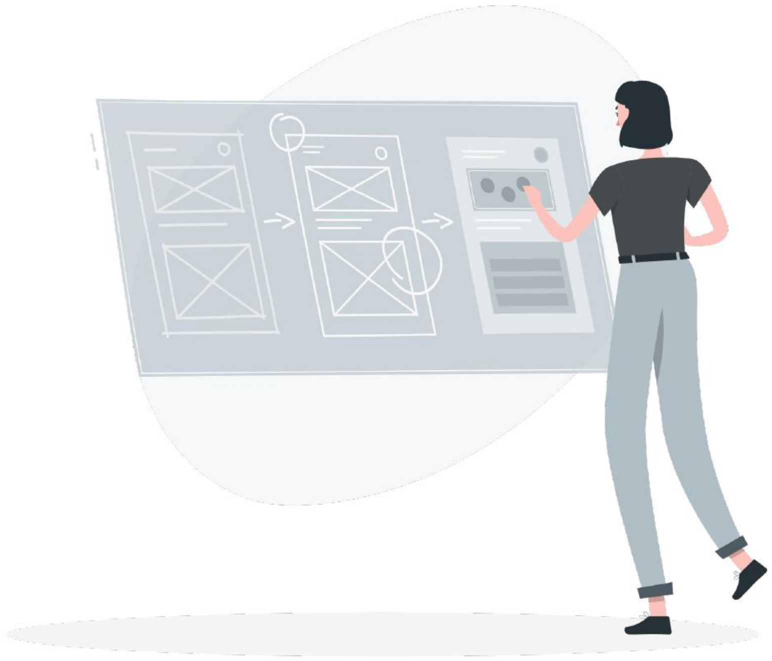
Dr. Abdelkarim Erradi

CSE@QU

Outline

1. Activity
2. Views
3. Constraint Layout

Activity

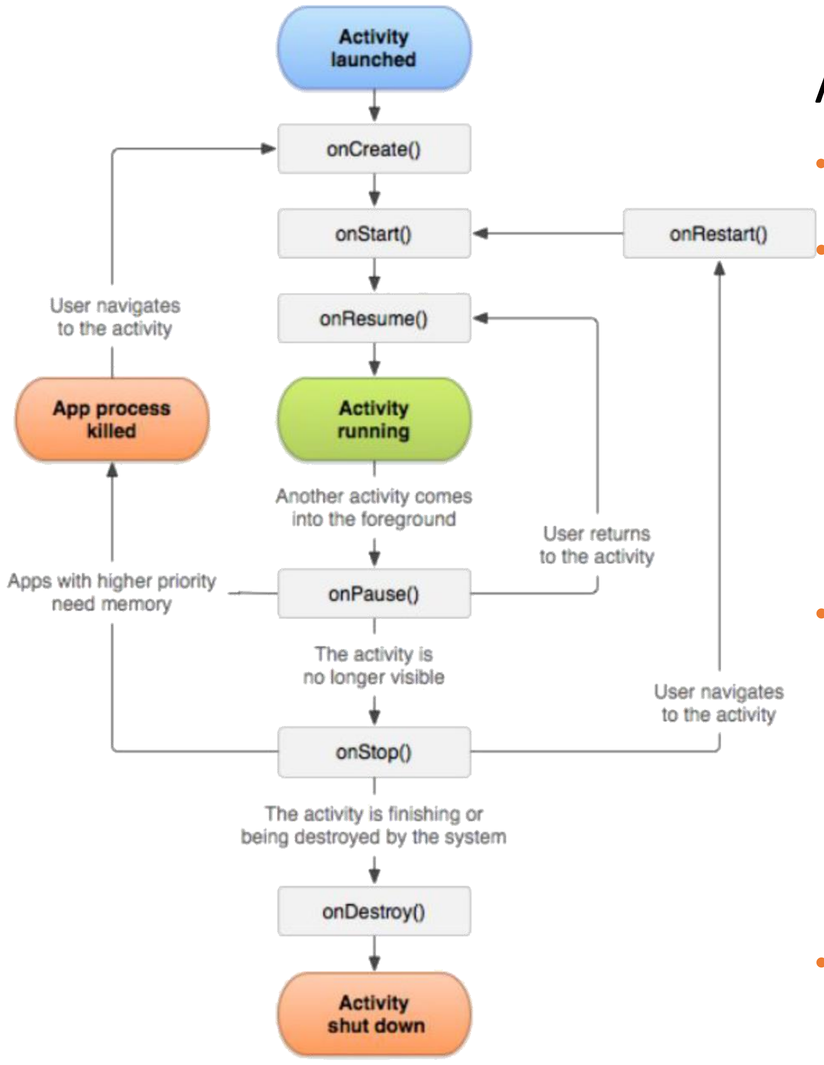


Back

Activity

- **Activity** provides the UI that the user interacts with.
  - Allow the user to do something such as order groceries, send email
  - Has layout (.xml) file & Activity class
  - This allows a **clear separation** between the UI and the app logic
- Connecting activity with the layout is done in the **onCreate** method
- Activity class define listeners to handle events:
  - User interaction events such press a button or enters text in a text view
  - External events such as receiving a notification or screen rotation
- Can start other activities in the same or other apps

Activity Lifecycle



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

An activity has essentially **four states**:

- ***Active*** if the activity in the foreground of the screen
- ***Paused*** if the activity has lost focus but is still visible (e.g., beneath a dialog box). A paused activity is alive but can be killed by the system in case of low memory.
  - When the user returns to the activity, it is **resumed**
- **Stopped** if the activity is completely obscured by another activity. It still retains its state but can be killed by the system when memory is needed.
  - When the user navigates to the activity, it must be **restarted** and restored to its previous state.
- **Destroyed** if an activity is paused or stopped, it maybe killed.
  - When the user navigates to the activity, it must be recreated.

Activity Lifecycle

onCreate()
Created but not yet active on the screen

→

onStart()
Activity is visible but not quite ready to receive focus or interaction

→

onResume()
Activity is visible and active in the app foreground

onPause()
Activity is visible, but something has taken foreground priority

→

onStop()
Activity is no longer visible on screen

→

onDestroy()
Activity is about to be destroyed because user navigated away and OS needs resources

Using Multiple Activities

- How do we **navigate** to a new screen?
  - Navigate to the new Activity using an **Intent**

```
val intent = Intent(this, RegisterActivity::class.java)
startActivity(intent)
```

- What is an Intent?
  - Communicates to the system that some action should be carried out
    - E.g., Start an Activity, send a tweet or an email, make a phone call
  - Implicit vs Explicit Intents: Choosing a generic actions vs specifying a specific app component
  - Specifies an **ACTION** and **DATA**

What is an Intent?

- Implicit intents describe an action like **`ACTION_SEND`** for sending an email
  - Implicit intents can be handled by a component in the system registered to handle that intent type. E.g.,
  - `ACTION_VIEW -> content://contacts/people/1`
  - `ACTION_DIAL -> content://contacts/people/1`
  - `ACTION_SEND -> EXTRA_EMAIL, EXTRA_SUBJECT`
- Explicit intents describe a specific app component to interact with
  - Can open a specific Activity using an explicit intent

Passing Data With Intents

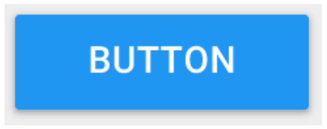
- Pass data

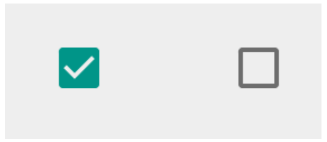
```
val intent = Intent(this, RegisterActivity::class.java)
// Pass student ID and student name with Intent so it can be
// used by RegisterActivity when it's started
intent.putExtra("id", 235789)
intent.putExtra("name", "Peter Pan")
startActivity(intent)
```

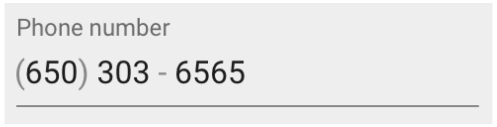
- Get passed data

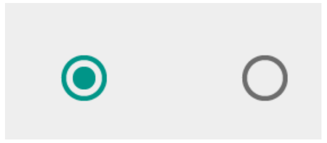
```
override fun onCreate(savedInstanceState: Bundle?) {
     ...
    // Read data sent by the caller
    val id = intent.getIntExtra("id", 0)
    val name = intent.getStringExtra("name")
}
```

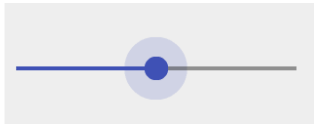
Views

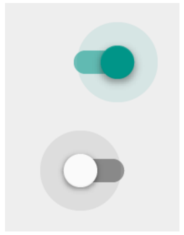
Button 

CheckBox 

EditText 

RadioButton 

SeekBar 

Switch 

< Back

Views

- **View = Widget = Control**
  - Examples: Button, Switch, Spinner, TextView, EditText, ImageView
  - Advanced Views (covered later): **RecyclerView** & MapView

- **Common Attributes**
  - id (i.e. **android:id="@+id/myViewId"**)
  - `layout_width`, `layout_height`
    - Values: `match_constraint` (or 0dp), `wrap_content`, fixed size (e.g., 50dp)

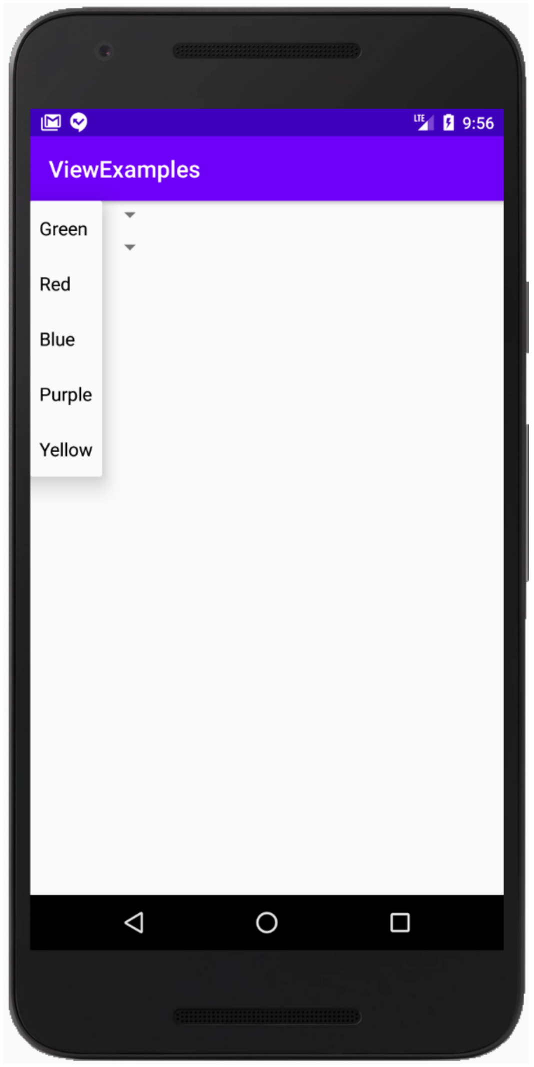
Views (Attributes and Listeners)

- TextView - Displays text on the screen
  - `text`
- EditText - Allows entering user input
  - `inputType  :` such as email, phone number, etc.
  - `text`
  - `.addTextChangedListener { … }`
- Button - Clickable view responding to user clicks
  - `text`
  - `.setOnClickListener { … }`
- ImageView - Displays image from a URL or from a resource file
  - `.setImageDrawable(drawable) // set image to display`
  - `.setOnClickListener { … }`

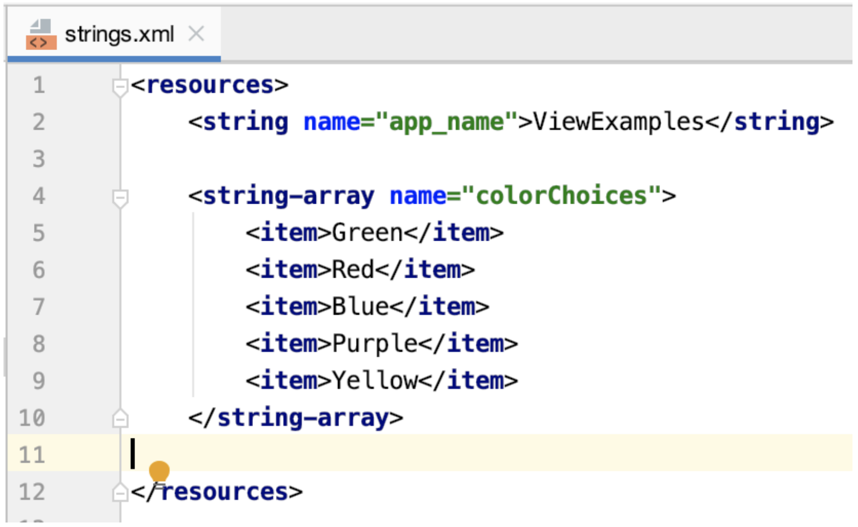
Views (Attributes and Listeners)

- **Switch (on/off)**
  - .checked = booleanVal – set check state
  - .setOnCheckedChangeListener { … }
- **Spinner (dropdown list)**
  - .setAdapter(ArrayAdapter) – specify list values
  - .setSelection(int) – specify selected item
  - onItemSelectedListener { … }
- **SearchView**
  - queryHint –text to display when the field is empty
  - iconifiedByDefault – Display the field or just an icon until clicked
  - .setIconified(boolean) – make always visible
  - .setOnQueryTextListener { … }

Setting Entries of a Spinner in the XML Layout File



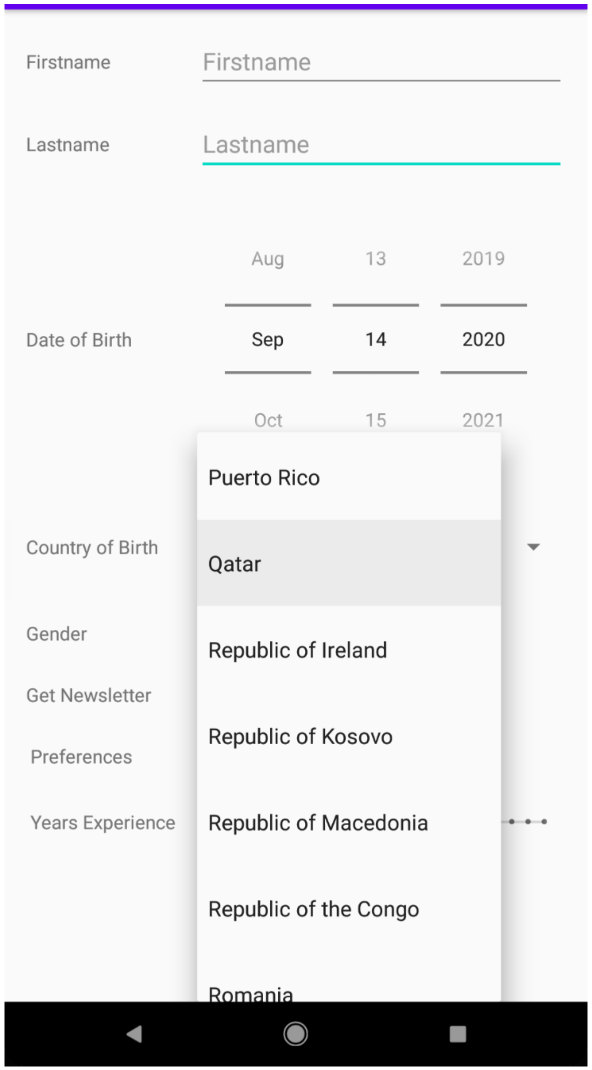
```xml
<Spinner
    android:id="@+id/colorSelector1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="32dp"
    android:layout_marginBottom="4dp"
    android:entries="@array/colorChoices"/>
```



strings.xml

```xml
<resources>
    <string name="app_name">ViewExamples</string>

    <string-array name="colorChoices">
        <item>Green</item>
        <item>Red</item>
        <item>Blue</item>
        <item>Purple</item>
        <item>Yellow</item>
    </string-array>

</resources>
```

Setting Entries of a Spinner in Code

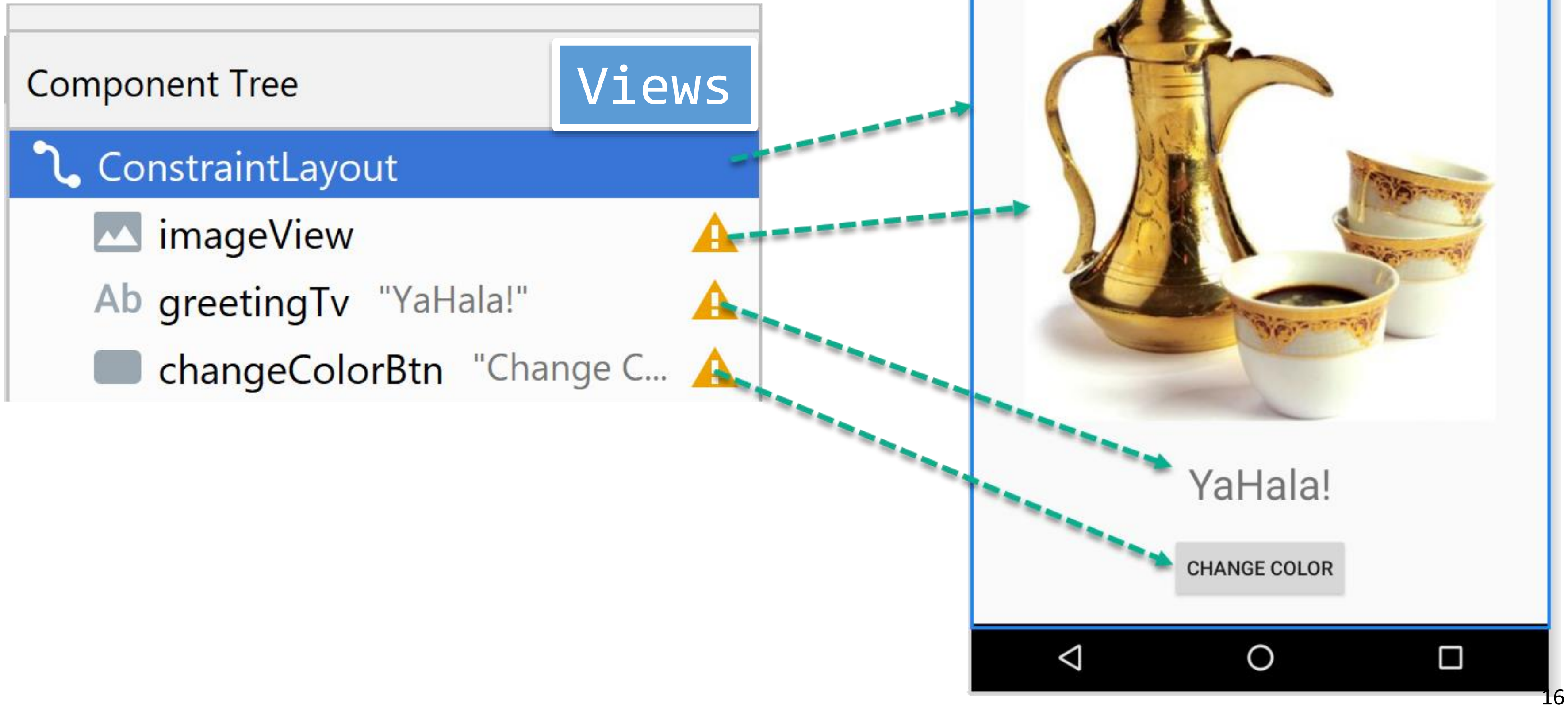

Firstname
Firstname
Lastname
Lastname
Date of Birth
Aug 13 2019
Sep 14 2020
Oct 15 2021
Country of Birth
Gender
Get Newsletter
Preferences
Years Experience
Puerto Rico
Qatar
Republic of Ireland
Republic of Kosovo
Republic of Macedonia
Republic of the Congo
Romania

```
<Spinner
    android:id="@+id/countriesSp"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
/>
```

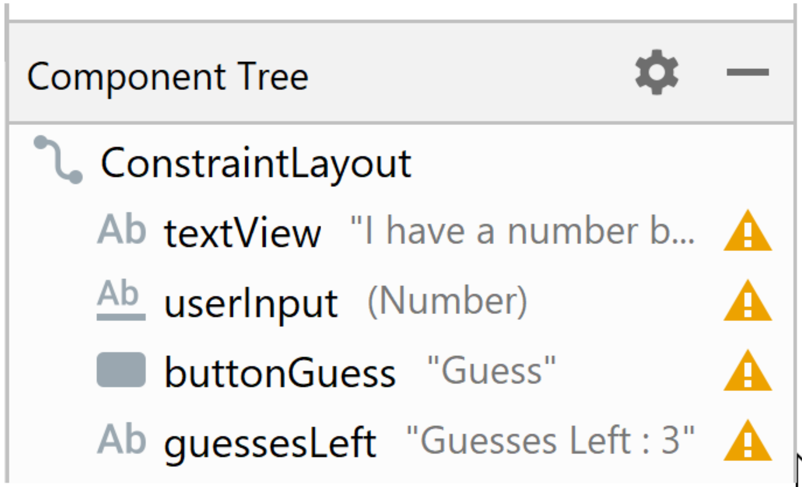
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_register)

    CountryRepository.loadCountries(this)

    val adapter = ArrayAdapter<String>(
        this,
        android.R.layout.simple_dropdown_item_1line,
        CountryRepository.countryNames
    )
    countriesSp.adapter = adapter
}
```

App 1 - Color Changer

App that contains Text reading "YaHala!", an Image and a Button that randomly changes text's color with every click



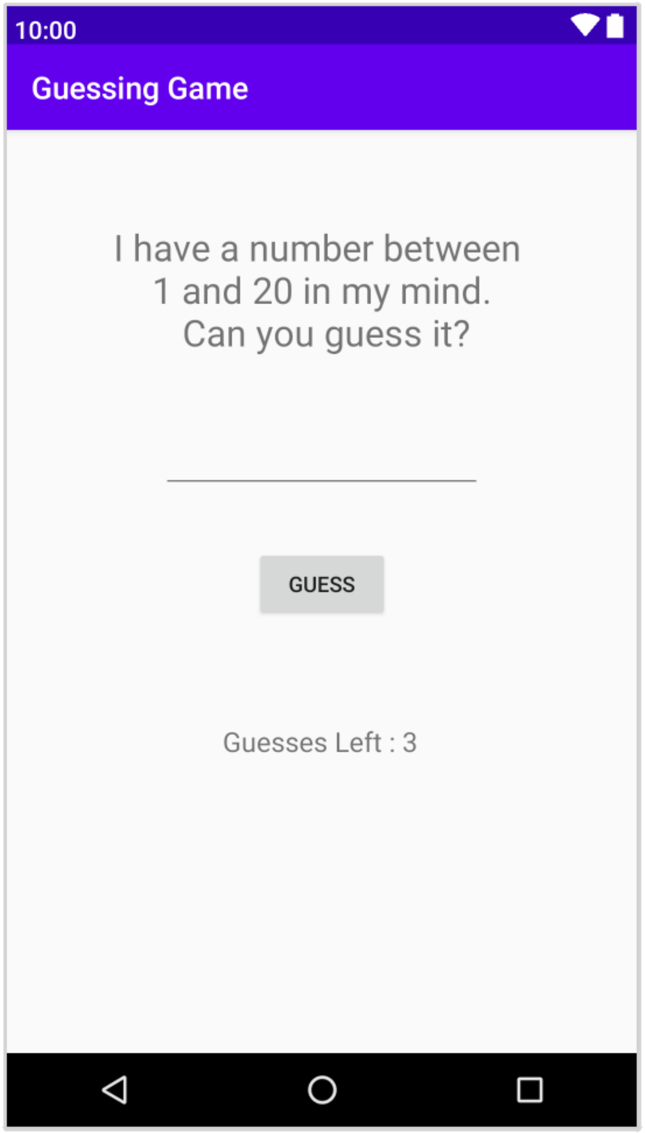
App 2 – Guessing Game



Component Tree

ConstraintLayout

- textView "I have a number b...
- userInput (Number)
- buttonGuess "Guess"
- guessesLeft "Guesses Left : 3"



10:00

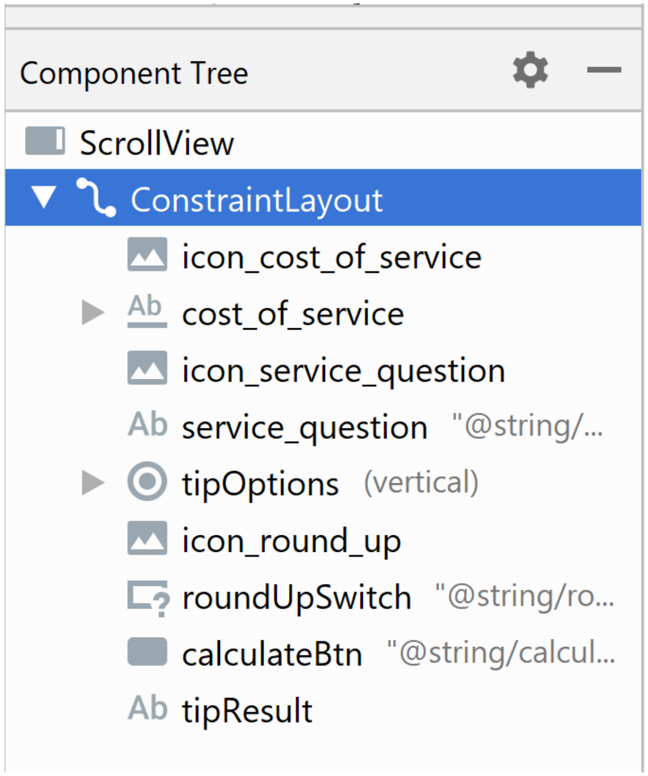
Guessing Game

I have a number between
1 and 20 in my mind.
Can you guess it?

GUESS

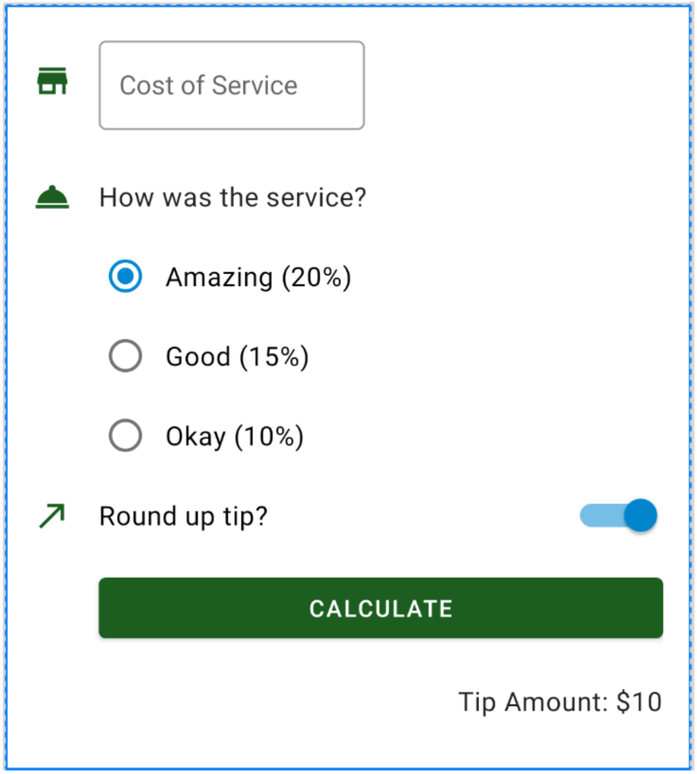
Guesses Left : 3

App 3 – Tips Calculator



Component Tree

- ScrollView
  - ConstraintLayout
    - icon_cost_of_service
    - cost_of_service
    - icon_service_question
    - service_question "@string/...
    - tipOptions (vertical)
    - icon_round_up
    - roundUpSwitch "@string/ro...
    - calculateBtn "@string/calcul...
    - tipResult



Cost of Service

How was the service?

(●) Amazing (20%)

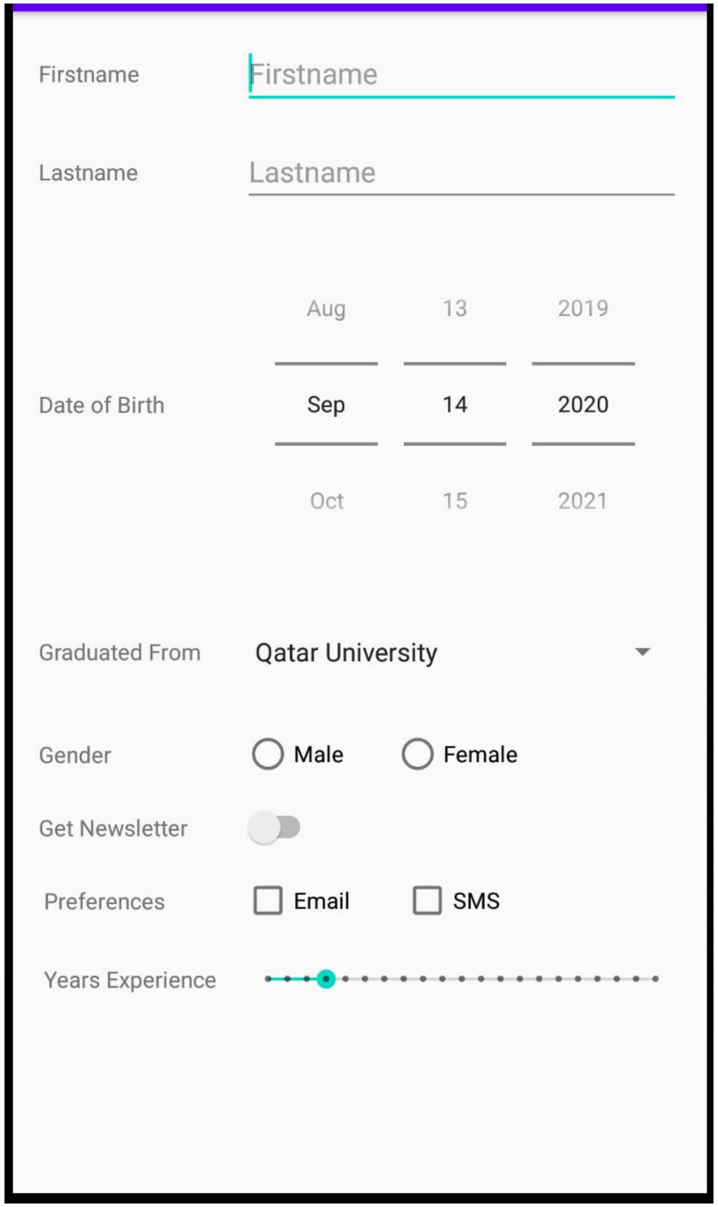
( ) Good (15%)

( ) Okay (10%)

Round up tip? [✓ on]

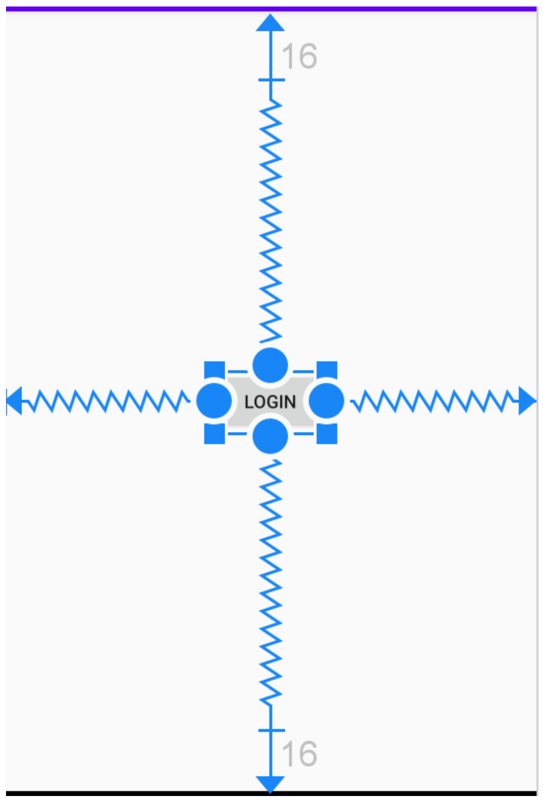
Tip Amount: $10

Registration Form



Firstname	Firstname
Lastname	Lastname
Date of Birth	Aug 13 2019 / Sep 14 2020 / Oct 15 2021
Graduated From	Qatar University
Gender	○ Male ○ Female
Get Newsletter	
Preferences	☐ Email ☐ SMS
Years Experience	

Constraint Layout



Back

Layouts

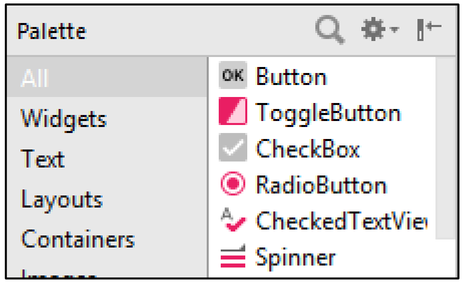
- Layout automatically **controls** the **size** and **placement** of views to create a **Responsive UI**
  - Frees programmer from handling/hardcoding the sizing and positioning of UI elements
  - **Responsive UI =** When the screen is resized, the views reorganize themselves based on the rules of the layout

Constraint Layout

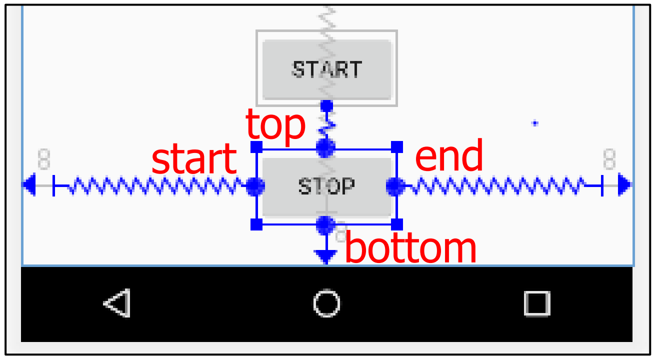
- ConstraintLayout: Allows buliding a Responsive UI by connecting views with constraints
  - Position a view relative others including the parent
  - Need to add at least one horizontal and one vertical constraint
  - Constraint is a connection to another view, parent layout, or invisible Guideline / Barrier
  - Uses constraints to determine the position and alignment of UI elements
  - Allows positioning UI elements in various ways: relative, centered, using **flow**

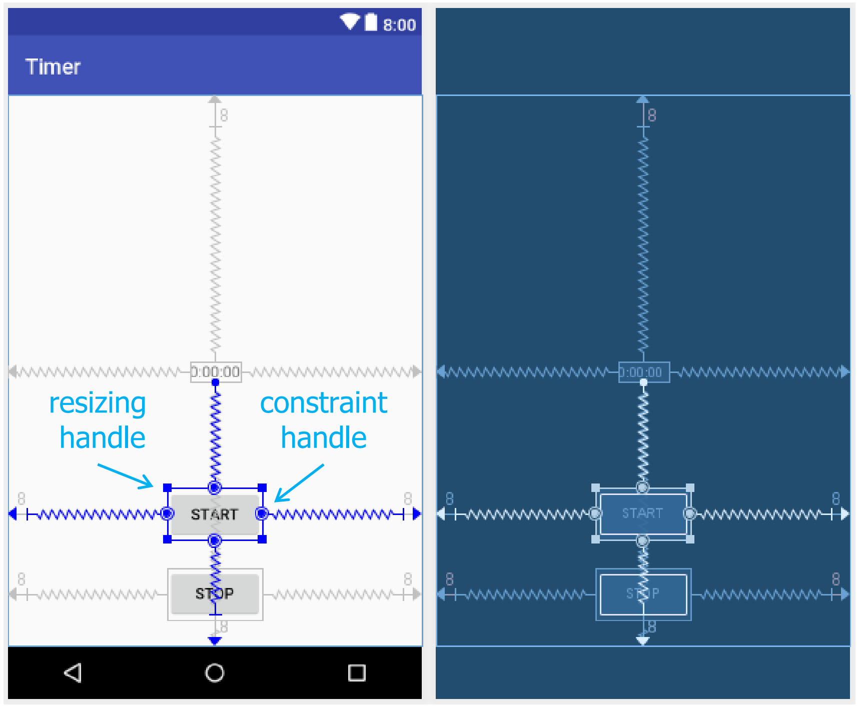
Defining Constraints

Steps
1. Drop a view to the editor
2. Connect constraint handles
   (e.g., top/bottom/left/right)



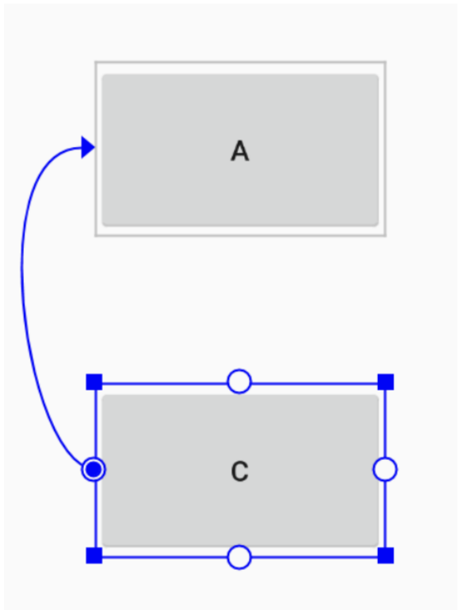
At least one horizontal and one vertical constraint





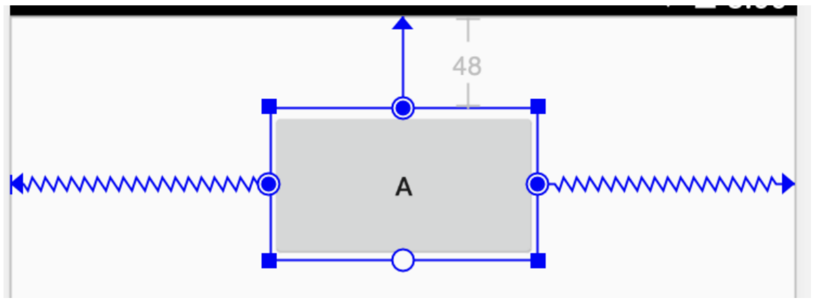
Alignment

- Align the edge of a view to the same edge of another view.
- The left side of C is aligned to the left side of A. If you want to align the view centers, create a constraint on both sides

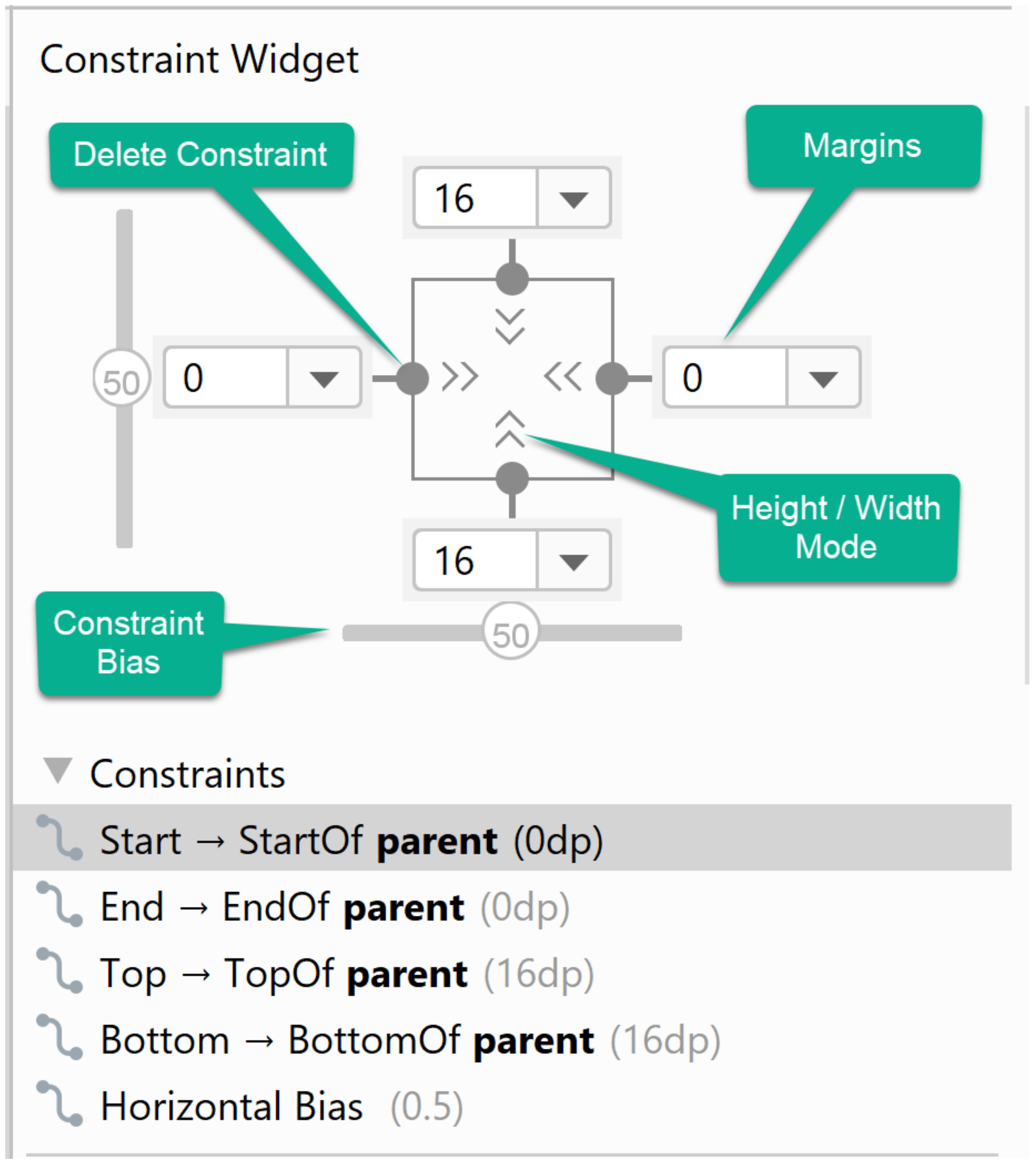


Bias

- If you add opposing constraints on a view, the constraint lines become like a **spring** to indicate the opposing forces.
- The view becomes centered between the two constraints with a bias of 50% by default.
- You can adjust the bias by dragging the view



View Constraints Editor



Constraints

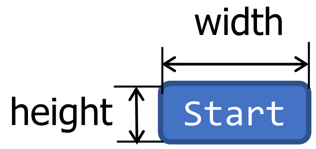
Start → StartOf **parent** (0dp)

End → EndOf **parent** (0dp)

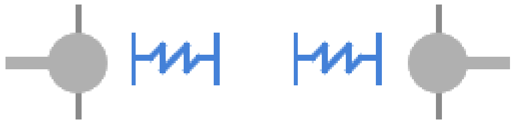
Top → TopOf **parent** (16dp)

Bottom → BottomOf **parent** (16dp)

Horizontal Bias (0.5)



View Size

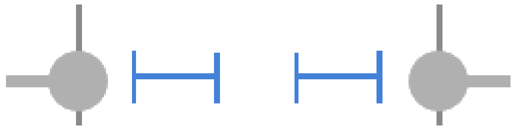


```
layout_width="0dp"
```

- The view expands to **match constraints** on each side (after accounting for the view's margins)
  - View will grow/shrink on resizing

```
layout_width="wrap_content"
```

- The view expands as needed to **fit** its contents

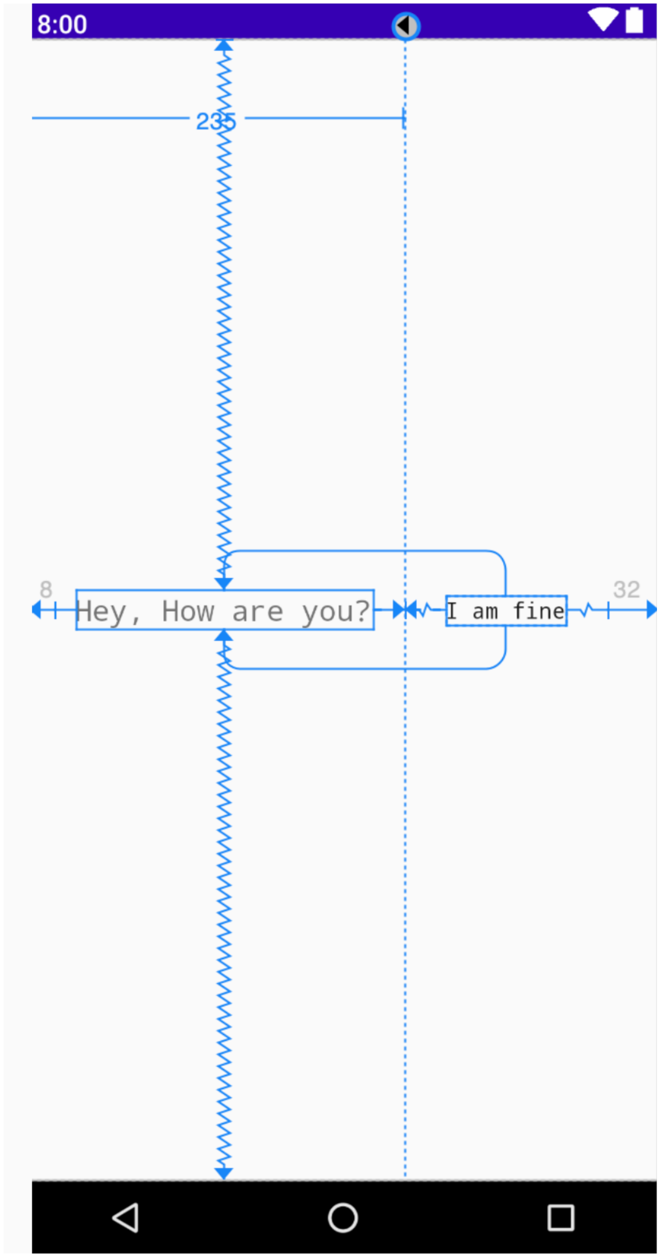


```
layout_width="200dp"
```

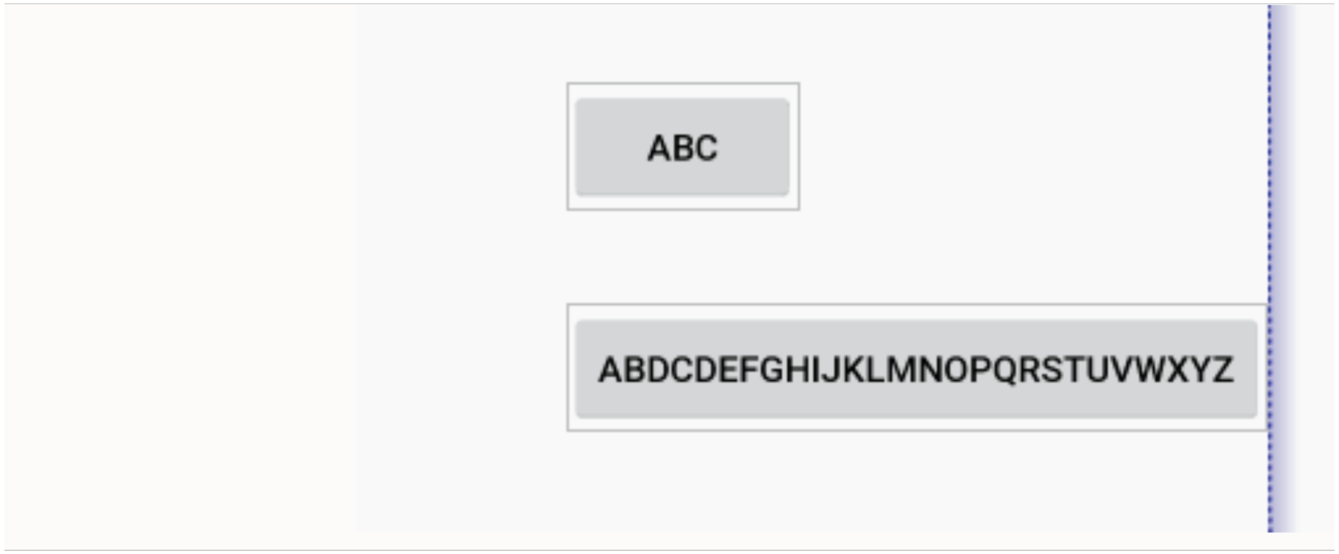
- **Fixed** size (e.g., 200dp density-independent pixels)

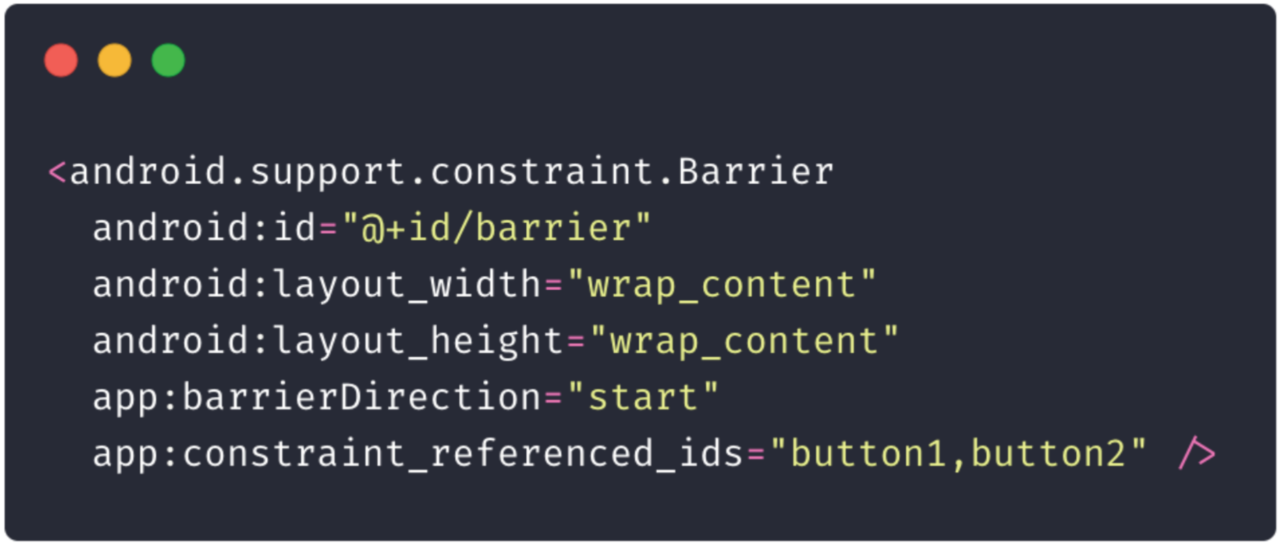
Guideline

- Add a vertical or horizontal **guideline** to which you can constrain views, and the guideline will be invisible to app users.
- Position the guideline within the layout based on either **dp** units or percent, relative to the layout's edge



Barrier

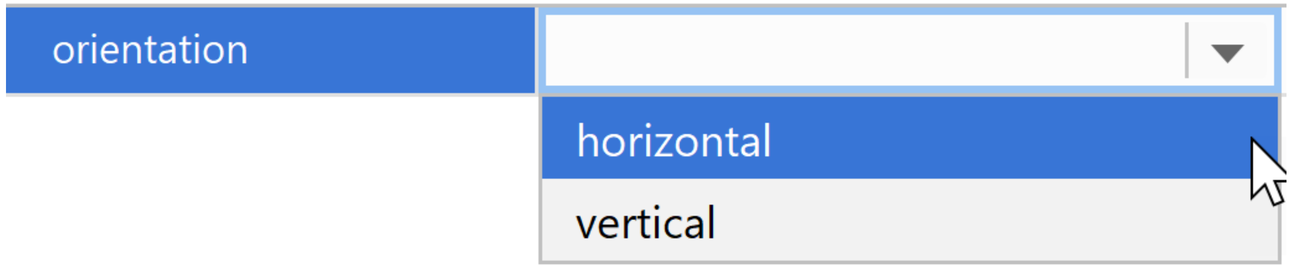


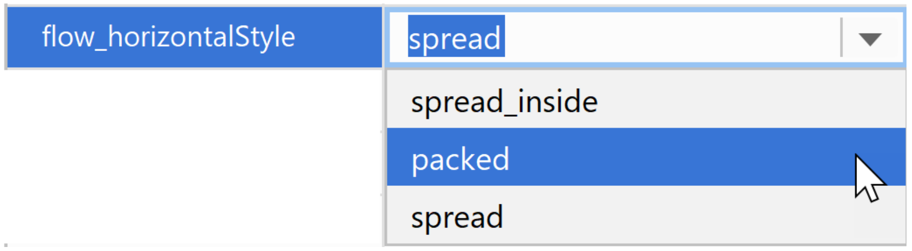


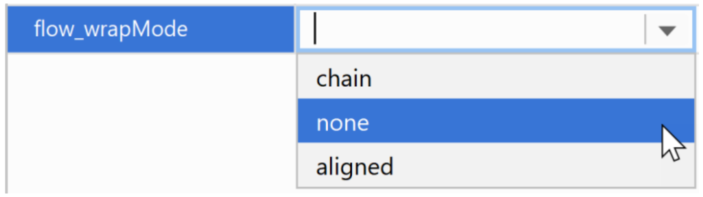
```
<android.support.constraint.Barrier
  android:id="@+id/barrier"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  app:barrierDirection="start"
  app:constraint_referenced_ids="button1,button2" />
```

Flow

- Flow provides an efficient way to distribute space among items in the flow while accommodating different screen sizes







Summary

- ConstraintLayout enables responsive design

.. mastering it will take some time and effort 🧗 🏋 …