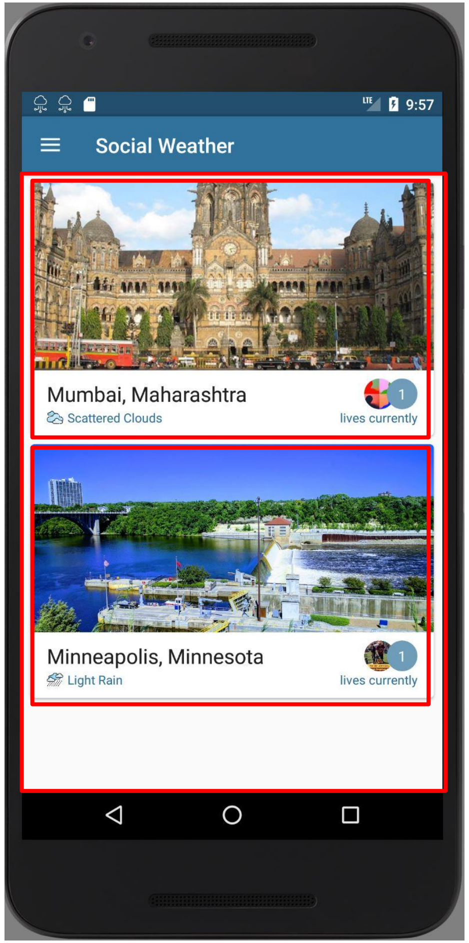
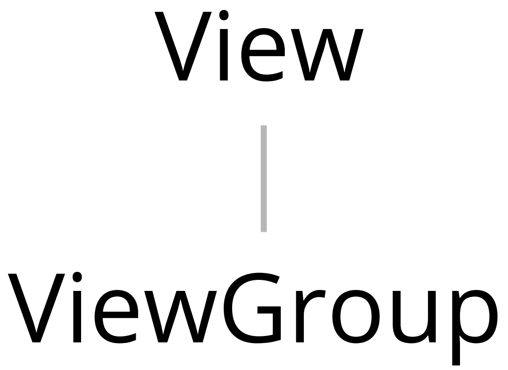
Android Week Two

Yehyun Ryu
UMN App Developers

ViewGroups

- Special type of views that can contain other views
- https://developer.android.com/reference/android/view/ViewGroup.html

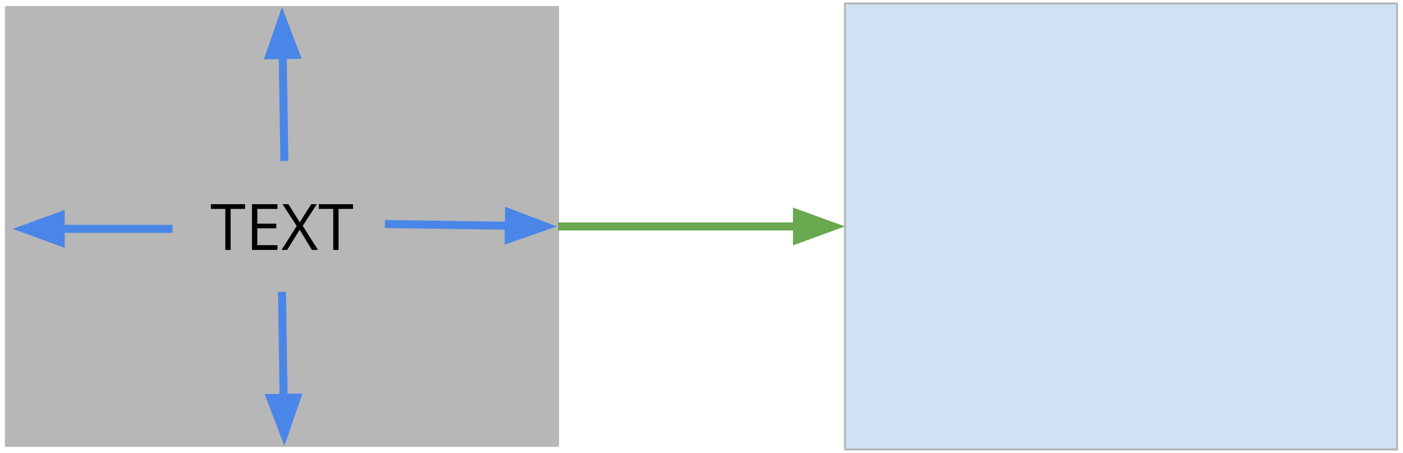




- A ViewGroup extends from a View
- A View is not necessarily a ViewGroup

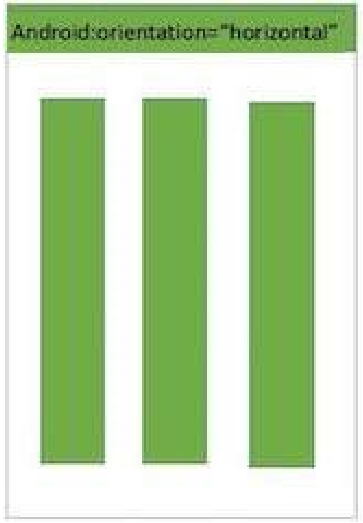
ViewGroups can contain other ViewGroups

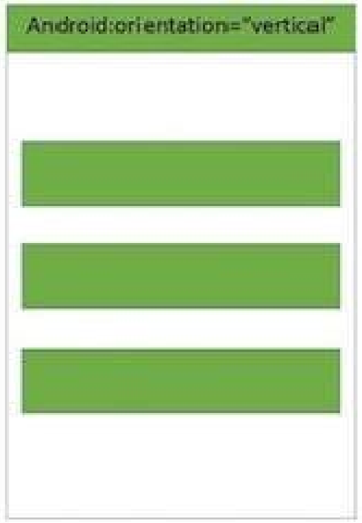
Margin and Padding



- Margin is distance from one view to another view
- Padding is the distance from inner content to outer shell of a view

LinearLayout





- Arranges children views in a certain orientation in a single row

LinearLayout Attributes

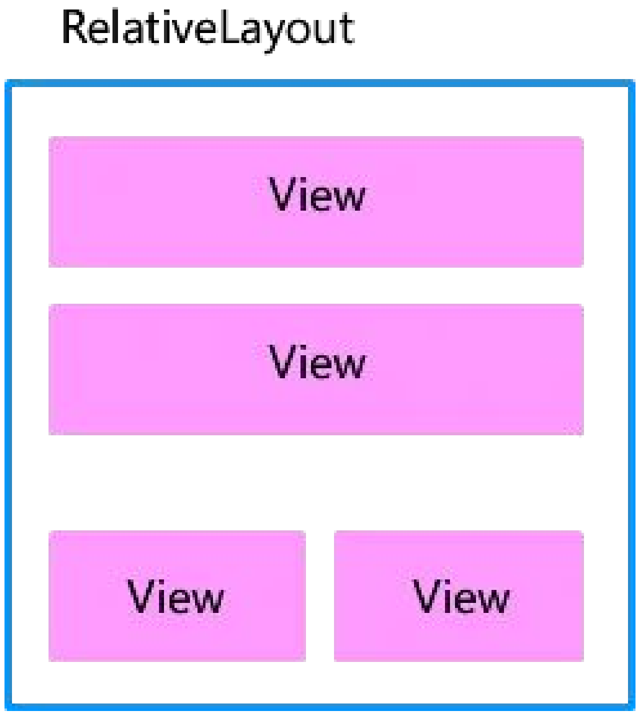
- Parent Attributes
  - android:orientation=”vertical” **OR** android:orientation=”horizontal”
  - vertical: displays children views from top to bottom
  - horizontal: displays children views from left ro right
- Children Attributes
  - android:layout_weight=”1”
  - (available space) * (current weight) / (total weight)
  - corresponds only with orientation
    - vertical: height
    - horizontal: width

gravity and layout_gravity

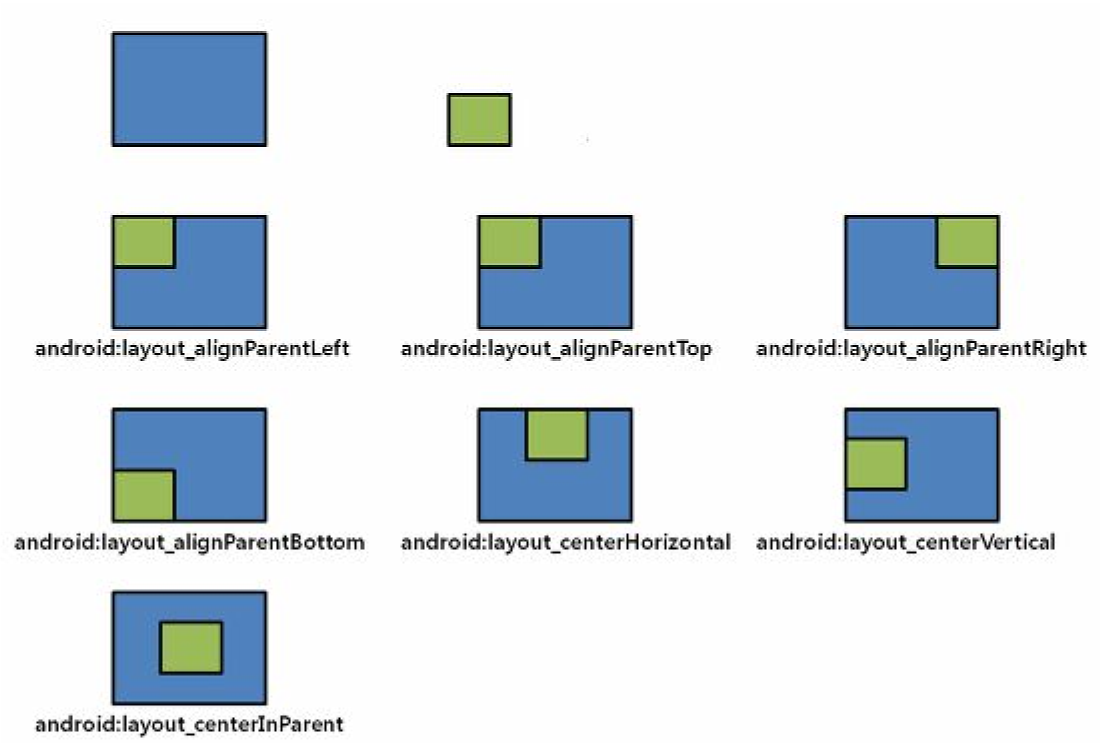
- gravity
  - Sets gravity of contents of a view
- layout_gravity
  - Sets gravity of view itself relative to parent

RelativeLayout

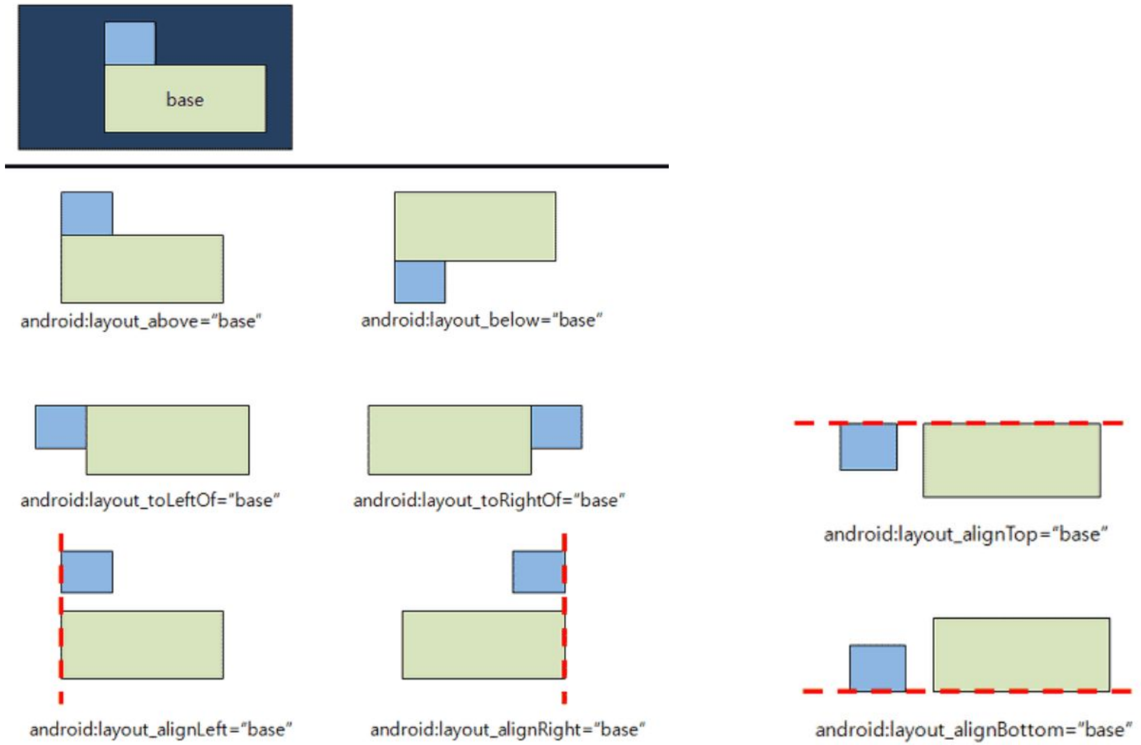
- Let's children views position itself relative to its parent (RelativeLayout) or to each other
- Remember to set an id attribute for each view



RelativeLayout Attributes (to Parent)

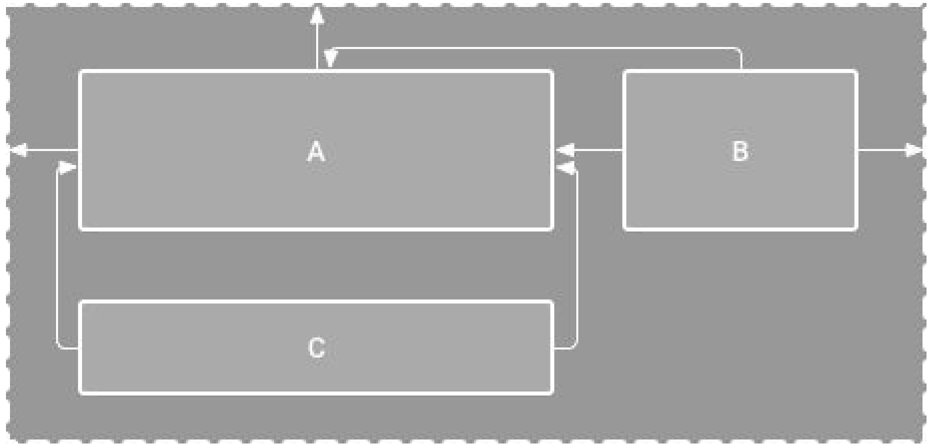

android:layout_alignParentLeft
android:layout_alignParentTop
android:layout_alignParentRight
android:layout_alignParentBottom
android:layout_centerHorizontal
android:layout_centerVertical
android:layout_centerInParent

RelativeLayout Attributes (to other Views)


base
android:layout_above="base"
android:layout_below="base"
android:layout_toLeftOf="base"
android:layout_toRightOf="base"
android:layout_alignTop="base"
android:layout_alignLeft="base"
android:layout_alignRight="base"
android:layout_alignBottom="base"

ConstraintLayout

- Allows children views to position and size itself in a very flexible way



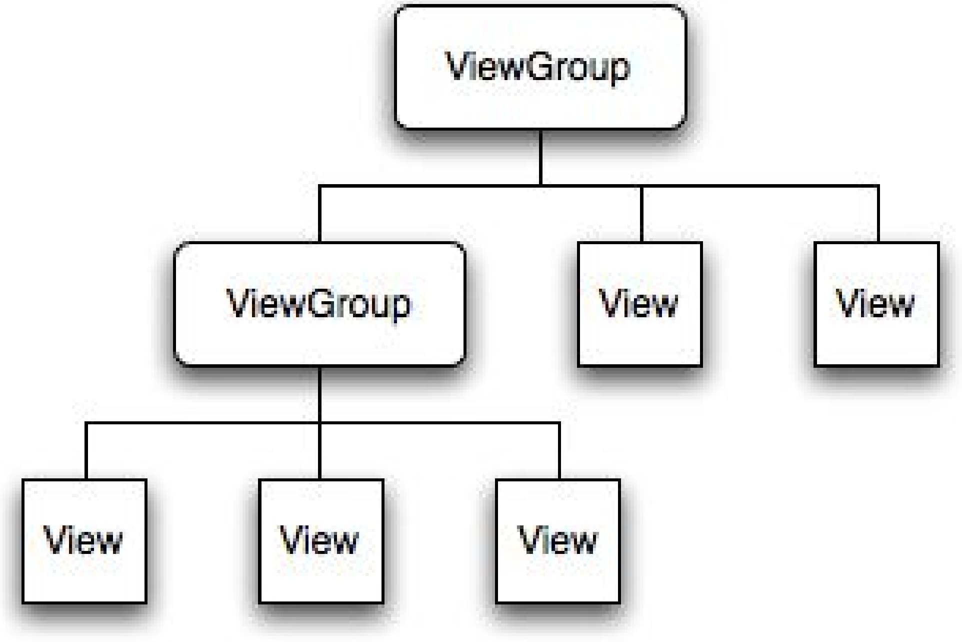
ConstraintLayout Attributes

- build.gradle (app)
  - Add ConstraintLayout Library
  - implementation **'com.android.support.constraint:constraint-layout:1.0.2'**
- Parent Attribute
  - Add app namespace
  - **xmlns:app="http://schemas.android.com/apk/res-auto"**
- Children Attributes
  - **app:layout_constraintTop_toTopOf="parent"**
  - **app:layout_constraintBottom_toBottomOf="parent"**
  - **app:layout_constraintLeft_toLeftOf="parent"**
  - **app:layout_constraintRight_toRightOf="parent"**

Why ConstraintLayout

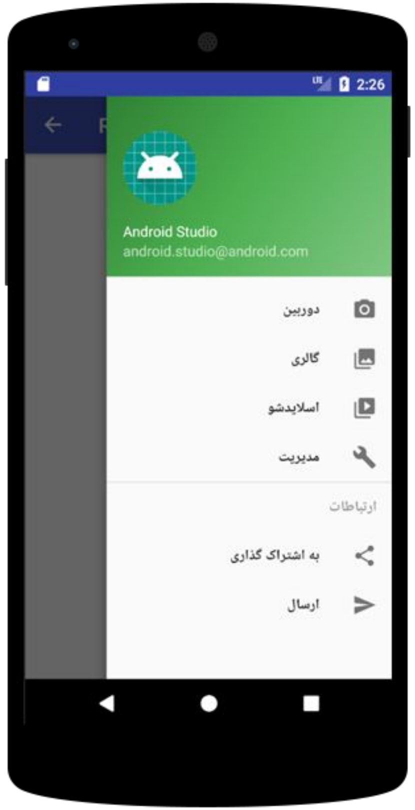
- Flexible
  - Can position views dynamically
  - Can chain views together
    - you only have to change one thing to make a small change
- More Efficient
  - Lower View Hierarchy
  - Takes less memory load
- LinearLayout
  - High View Hierarchy => More memory
- RelativeLayout
  - Not flexible
  - ex) position a view between two other views is a nightmare

View Hierarchy



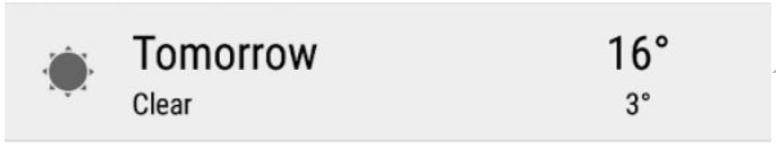
Right-To-Left Support

- Some languages and cultures read from right-to-left instead of left-to-right
- Android supports this by automatically changing the direction of a layout
- left = start
- right = end

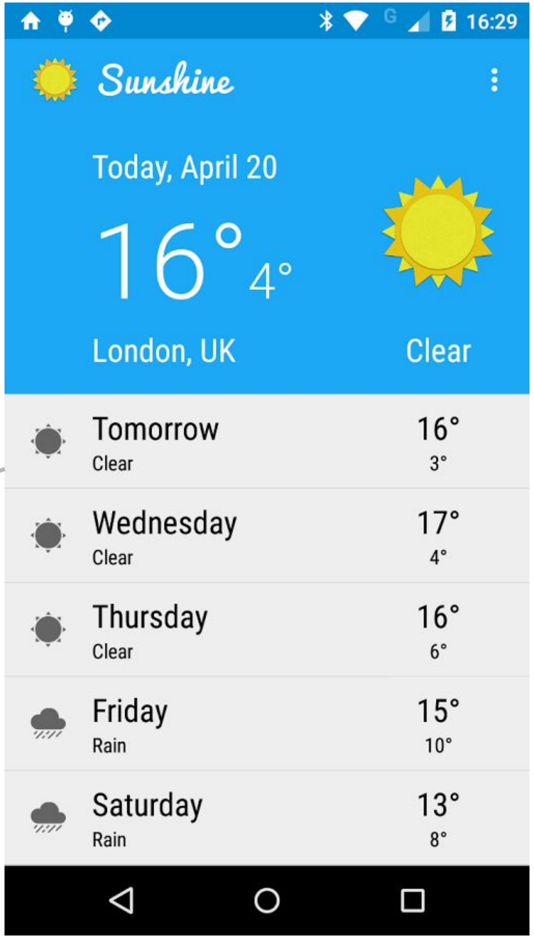


Today's Code

- Experiment with different viewgroup for a single weather item layout



Tomorrow 16°
Clear 3°



16:29

Sunshine

Today, April 20

16° 4°

London, UK Clear

Tomorrow	Clear	16° / 3°
Wednesday	Clear	17° / 4°
Thursday	Clear	16° / 6°
Friday	Rain	15° / 10°
Saturday	Rain	13° / 8°