

# Google Developer Challenge Scholarship

Course notes & summary

# SUMMARY

P.3	• VIEWS
P.4	• XML SYNTAX
P.5	• ATTRIBUTES
P.14	• VIEW GROUPS
P.15	• RELATIVE LAYOUT
P.23	• LINEAR LAYOUT
P.30	• CONSTRAINT LAYOUT
P.31	• PADDING VS. MARGIN
P.34	• PROBLEM SOLVING

# VIEWS

## **PARENT VIEWS** / VIEW GROUPS / ROOT VIEWS

set the rules for positioning ChildrenViews within them

- RelativeLayout
- LinearLayout
- ConstraintLayout (to build responsive UIs)

## **CHILDREN VIEWS**

- TextView
- ImageView
- Button
- ...

# XML SYNTAX

ELEMENT	CODE	EXAMPLE	NOTES
PARENT XML ELEMENT	<ViewGroupName	E.g. <RelativeLayout <LinearLayout <ConstraintLayout	CamelCase format: no space between words, 1 <sup>st</sup> letter of each word is capitalized
<i>separate opening tag</i>	...attributes >		(attribute:name="attribute value")
CHILD XML ELEMENT	<ChildView	E.g. <TextView <ImageView <Button	
<i>self-closing tag (usually for CHILDREN XML ELEMENTS)</i>	...attributes />		
<i>separate closing tag</i>	</ ViewGroupName>		

# ATTRIBUTES

## SIZE OF VIEWS

### FIXED DP VALUES

200 dp



`android:layout_width="...dp"`

`android:layout_height="...dp"`

- not adapting to all devices: content might be cut off
- dp= density-independent pixels - unit of measure describing the size of Views on Android & the distance in between Views - same physical size across devices, but Android is mapping it to a different number of pixels
- make touch targets 48dp at least (approx. 9 mm)  
<https://material.io/guidelines/layout/metrics-keylines.html#metrics-keylines-touch-target-size>

200 dp



wrap-content

### WRAP CONTENT



`android:layout_width="wrap_content"`

`android:layout_height="wrap_content"`

- adjusts width/height of the View so it's just as wide/tall as the content inside of it

wrap-content



match-parent

### MATCH PARENT



`android:layout_width="match_parent"`

`android:layout_height="match_parent"`

- width/length of each ChildView is as wide/tall as the parent ViewGroup, regardless of the content inside of it

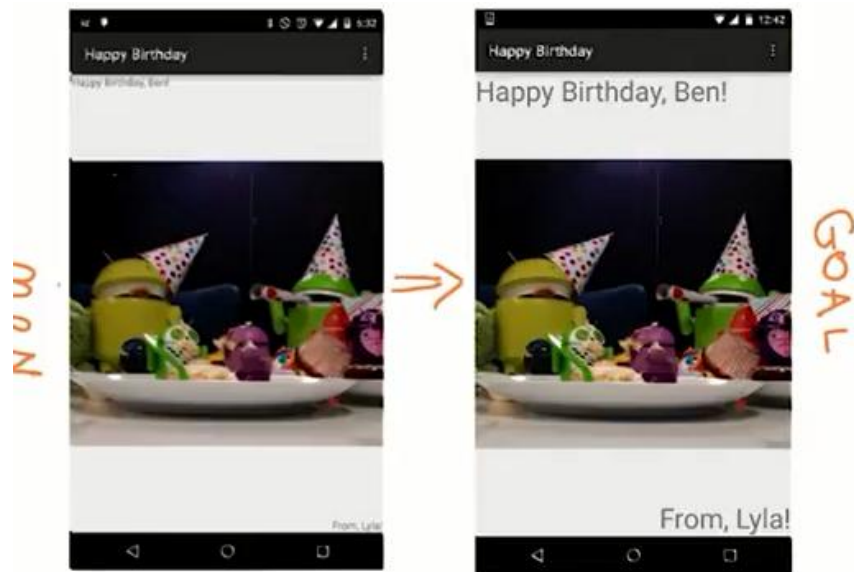
match-parent



# ATTRIBUTES

## TEXT SIZE

`android:textSize="...sp"`



- Scaled- Pixels: unit of measure similar to dp which makes the app look consistent across different devices
- Recommended text sizes (best to pick a couple of styles only)  
<https://material.io/guidelines/style/typography.html#typography-styles>

# ATTRIBUTES

## TEXT APPEARANCE – STANDARD TEXT SIZE

**`android:textAppearance="?android:textAppearanceLarge"`**

To be consistent with other apps on the platform, a standard set of type sizes can be used: small, medium, or large.

As of API 19 (KitKat):

- *?android:textAppearanceSmall* is currently 14sp
- *?android:textAppearanceMedium* is currently 18sp
- *?android:textAppearanceLarge* is currently 22sp

<https://plus.google.com/+AndroidDevelopers/posts/gQuBtnuk6iG>

# ATTRIBUTES

## TEXTVIEW ALL CAPS – USUALLY USED FOR BUTTONS

Leave the text as is and toggle this attribute to change it from “true” to “false”

- `android:textAllCaps="true"`
- `android:textAllCaps="false"`



# ATTRIBUTES

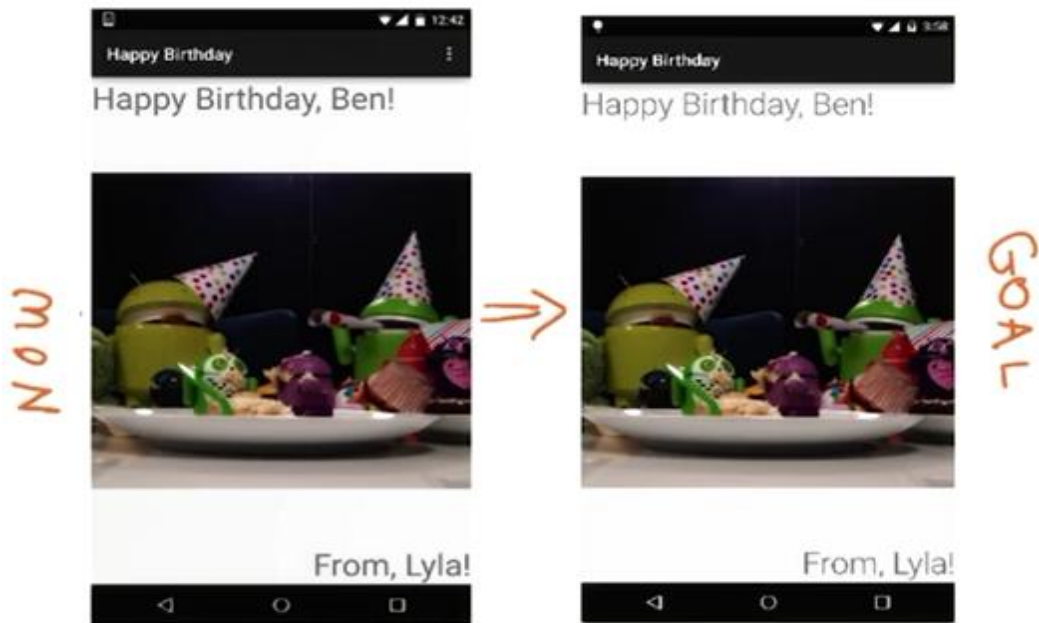
## TEXT STYLE - BOLD OR ITALIC

- `android:textStyle="bold"`
- `android:textStyle="italic"`
- `android:textStyle="bold|italic"`

# ATTRIBUTES

## TEXT FONT – fontFamily

`android:fontFamily="..."`



# ATTRIBUTES

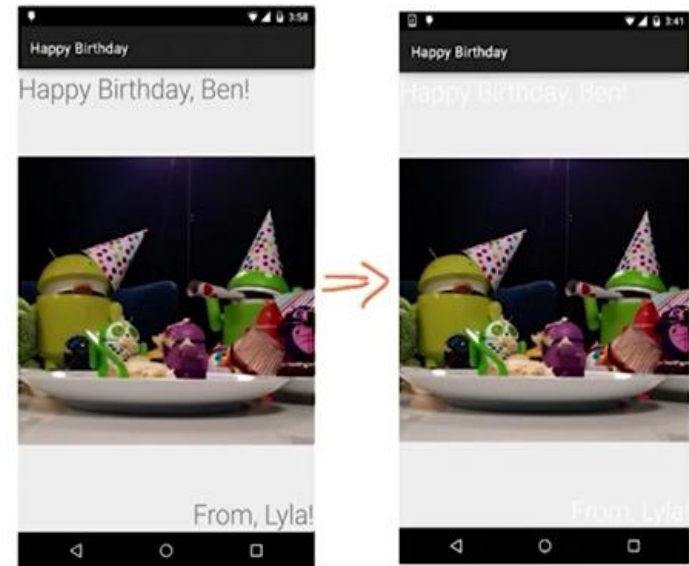
## BACKGROUND/TEXT COLOR

- `android:background="@android:color/..."` (darker\_gray – black – white)

or `android:background="#..."`

- `android:textColor="@android:color/..."`

or `android:textColor="#..."`



For “#...” specify the exact Hex color code

<https://material.io/guidelines/style/color.html#color-color-palette>

[https://www.w3schools.com/colors/colors\\_hex.asp](https://www.w3schools.com/colors/colors_hex.asp)

# ATTRIBUTES

## IMAGEVIEW

- **android:src="@drawable/..file name.."**

add images manually into the drawable folder of an app

- **android:scaleType="center"**

doesn't change size of the image, just centers it

- **android:scaleType="centerCrop"**

– centers image and crops it to fit the height and width of the View

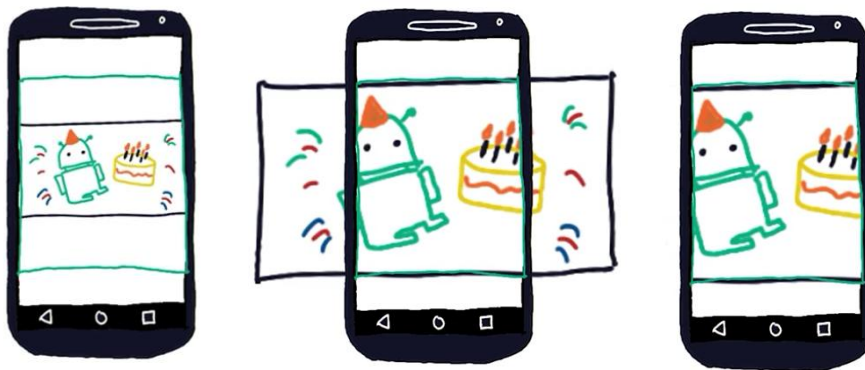
– maintains the aspect ratio of the original image so that it doesn't get distorted

– best for displaying photographs to achieve edge to edge look and give it a nice immersive feel

<http://labs.udacity.com/android-visualizer/#/android/simple-imageview>

# ATTRIBUTES

## IMAGEVIEW – `scaleType="centerCrop"`



CenterCrop

The documentation says

“Scale image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding)”

- `scaleType`, `layout_width` and `layout_height` all work together
- `centerCrop` will scale this image up so that it fits in the box (actual size of the `ImageView`), since it expands over the edge of the phone it will crop off the excess parts of the image

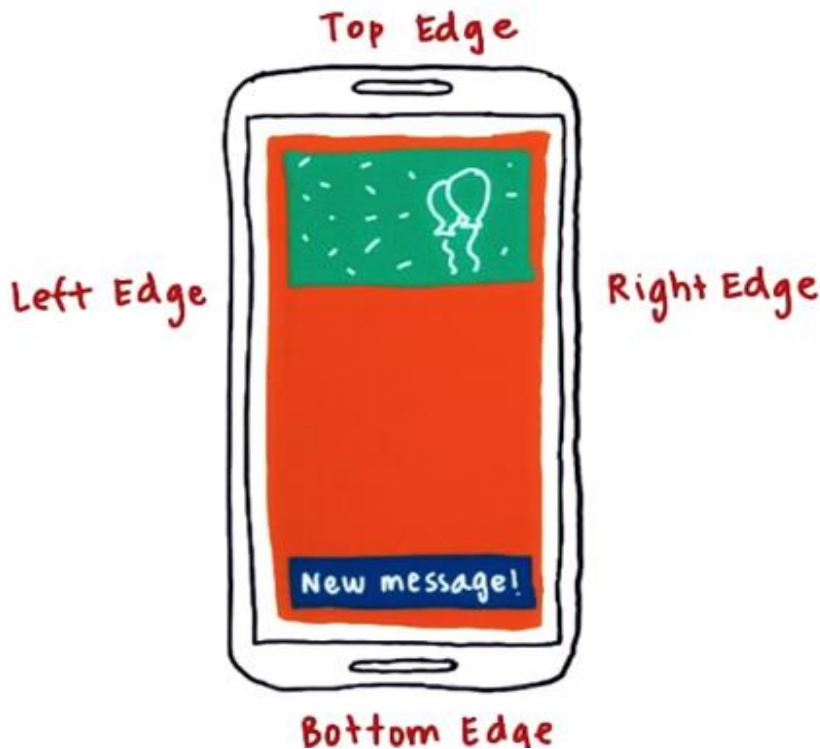
# VIEW GROUPS

- RelativeLayout
  - Position ChildrenViews relative to the ParentView
    - ChildView aligned to the bottom of the parent
    - ChildView aligned to the top of the parent
    - ChildView aligned to the parent's right edge
    - ChildView aligned to the parent's left edge
    - ChildView aligned to the center
  - Position a ChildrenViews relative to other ChildrenViews
- LinearLayout
  - Arrange its ChildrenViews in a vertical column
  - Arrange its ChildrenViews in a horizontal row
- ConstraintLayout

[https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html?utm\\_source=udacity&utm\\_medium=course&utm\\_campaign=android\\_basics](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics)

# RelativeLayout

- Relative to Parent



ImageView attributes:

```
android:layout_alignParentTop="true"
```

```
android:layout_alignParentBottom="false"
```

```
android:layout_alignParentLeft="true"
```

```
android:layout_alignParentRight="true"
```

TextView attributes:

```
android:layout_alignParentTop="false"
```

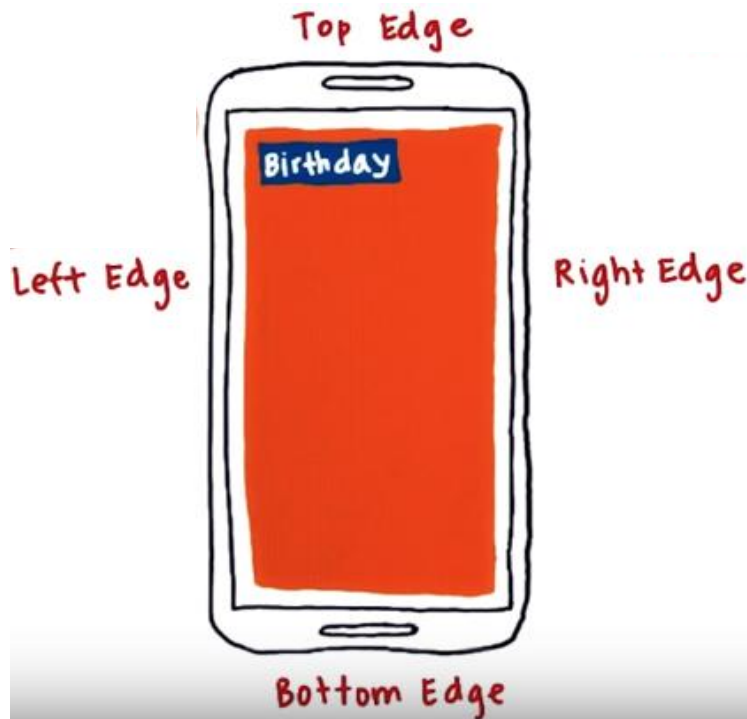
```
android:layout_alignParentBottom="true"
```

```
android:layout_alignParentLeft="true"
```

```
android:layout_alignParentRight="true"
```

# RelativeLayout

- Relative to Parent – Top Left Edge



child view attributes:

`android:layout_alignParentTop = "true"`

`android:layout_alignParentBottom`

`android:layout_alignParentLeft = "true"`

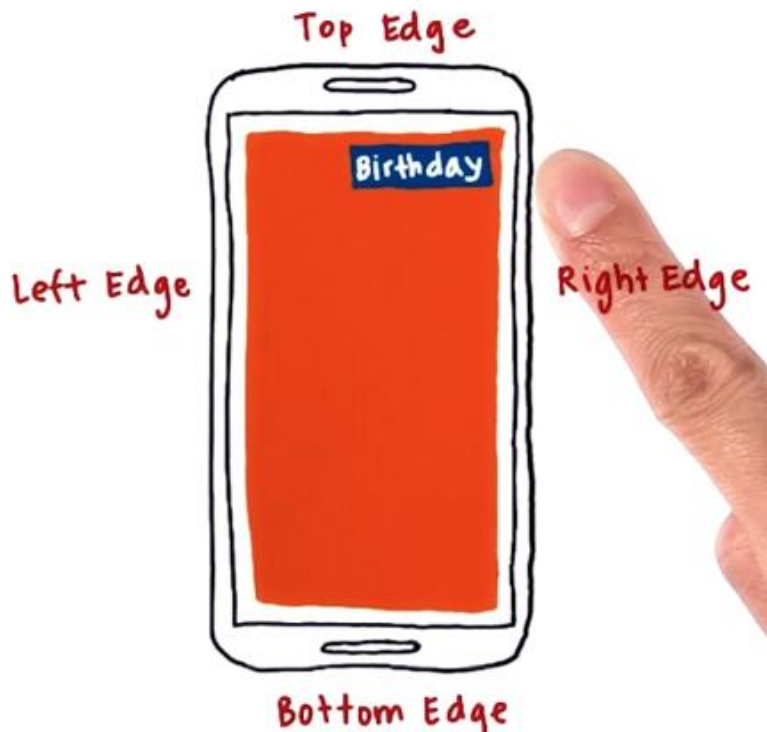
`android:layout_alignParentRight`

- By default positioned in the top left corner
- It is possible to specify just the "true" values as "false" values are default values



# RelativeLayout

- Relative to Parent – Top Right Edge



Child view attributes:

`android:layout_alignParentTop = "true"`

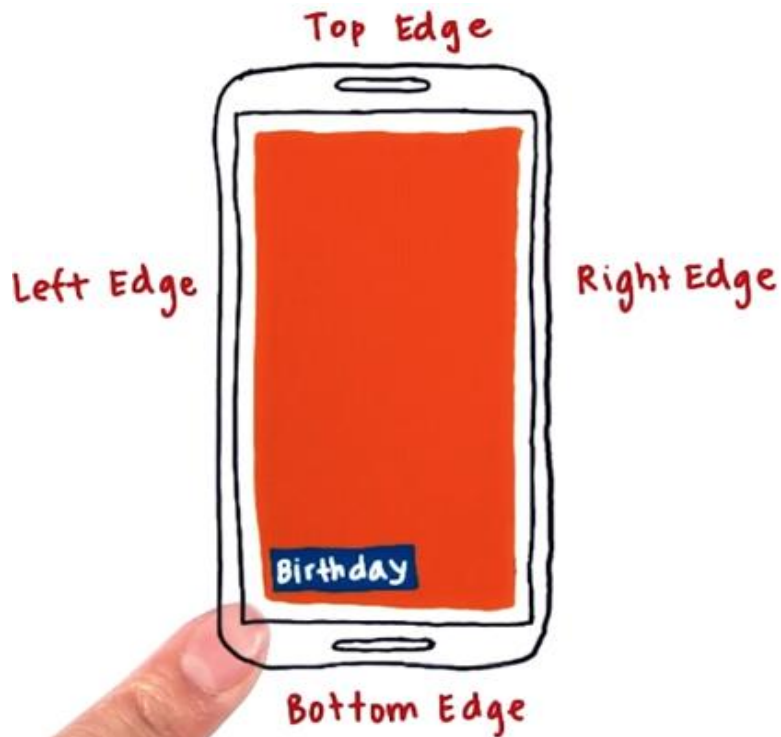
`android:layout_alignParentBottom`

`android:layout_alignParentLeft`

`android:layout_alignParentRight = "true"`

# RelativeLayout

- Relative to Parent – Bottom Left Edge



Child view attributes:

`android:layout_alignParentTop`

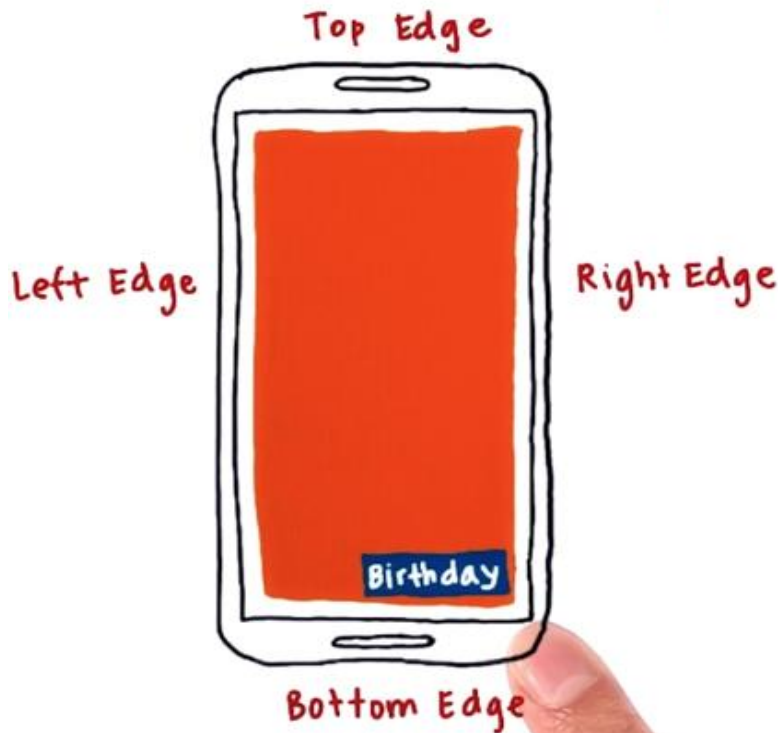
`android:layout_alignParentBottom = "true"`

`android:layout_alignParentLeft = "true"`

`android:layout_alignParentRight`

# RelativeLayout

- Relative to Parent – Bottom RightEdge



child view attributes:

`android:layout_alignParentTop`

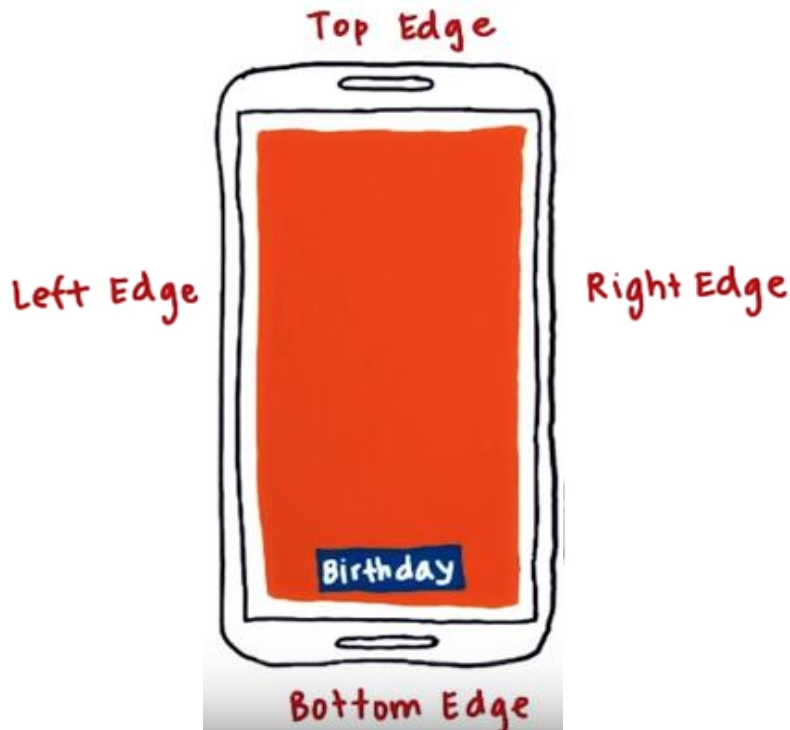
`android:layout_alignParentBottom = "true"`

`android:layout_alignParentLeft`

`android:layout_alignParentRight = "true"`

# RelativeLayout

- Relative to Parent – CenterHorizontal



Child view attributes:

`android:layout_alignParentTop`

`android:layout_alignParentBottom = "true"`

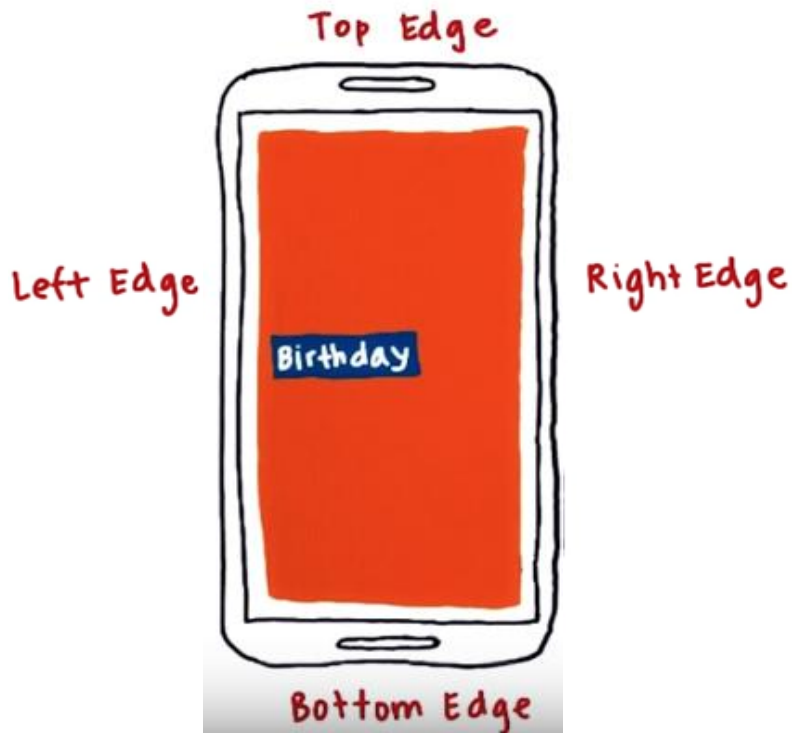
`android:layout_alignParentLeft`

`android:layout_alignParentRight`

`android:layout_centerHorizontal = "true"`

# RelativeLayout

- Relative to Parent – CenterVertical



Child view attributes:

`android:layout_alignParentTop`

`android:layout_alignParentBottom`

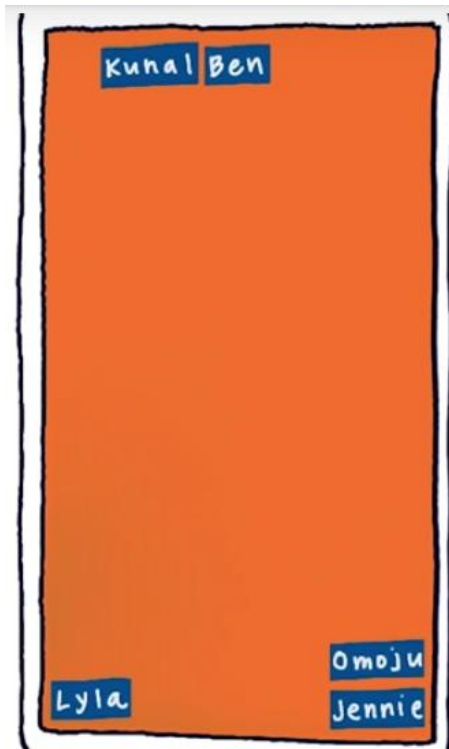
`android:layout_alignParentLeft`

`android:layout_alignParentRight`

`android:layout_centerVertical = "true"`

# RelativeLayout

- Relative to other ChildrenViews



## ➡ Assigning view ID names

On Ben TextView  
`android:id="@+id/ben-text-view"`

## ➡ Positioning children relative to other views

On Kunal TextView:  
`android:layout_toLeftOf =  
"@id/ben-text-view"`

## ➡ Assigning view ID names

On Jennie TextView:  
`android:id="@+id/jennie-text-view"`

## ➡ Positioning children relative to other views

On Omoju TextView:  
`android:layout_above =  
"@id/jennie-text-view"`

- "@+" to define a new ID
- no spaces, needs to start with a letter, can contain letters and numbers, no symbols
- XML Name Space Declaration to prevent name conflicts where 2 attributes are named the same thing but actually have different behaviors

`xmlns:android="..http://schemas.android.com/apk/res/android.."`

[https://www.w3schools.com/xml/xml\\_namespaces.asp](https://www.w3schools.com/xml/xml_namespaces.asp)

# LinearLayout



`android:orientation="vertical"`

`android:orientation="horizontal"` (default)



# LinearLayout

- **LinearLayout Weight** – evenly spacing out ChildrenViews

What we have:



`android:layout_weight="..."`

and

What we want:

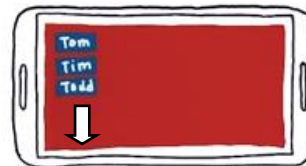
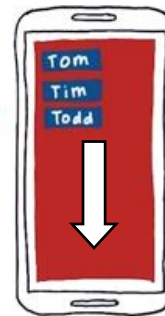


`android:layout_width="0dp"`

or

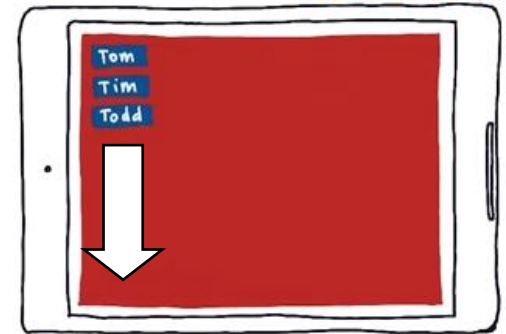
`android:layout_height="0dp"`

Portrait  
Mode



TAKING ADVANTAGE OF  
SCREEN REAL ESTATE

Tablet Landscape



Landscape Mode

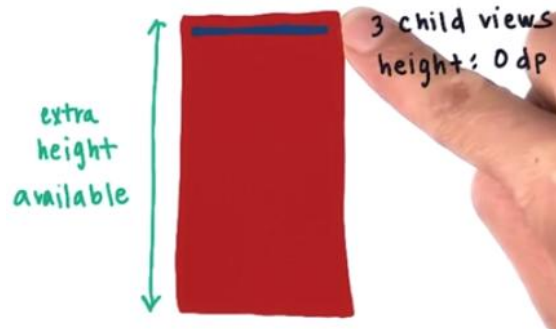
- Assigns an "importance" value to a View in terms of how much space it should occupy on the screen.
- Any remaining space in the ViewGroup is assigned to ChildrenViews in the proportion of their declared weight.

[https://developer.android.com/guide/topics/ui/layout/linear.html?utm\\_source=udacity&utm\\_medium=course&utm\\_campaign=android\\_basics](https://developer.android.com/guide/topics/ui/layout/linear.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics)  
<https://stackoverflow.com/questions/3470420/is-it-possible-to-evenly-distribute-buttons-across-the-width-of-an-android-linea>

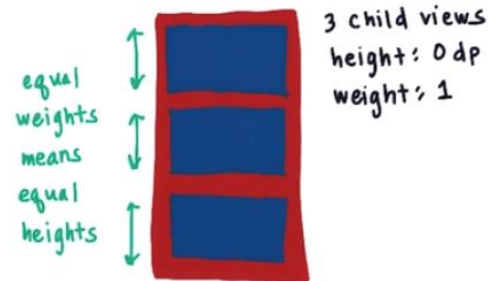


# LinearLayout

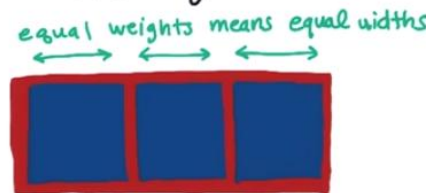
- LinearLayout Weight



Vertical  
LinearLayout



Horizontal  
Linear layout



3 child views  
width: 0 dp

3 child views  
width: 0 dp  
weight: 1

# LinearLayout

- LinearLayout Weight



Vertical Linear Layout

	Width	Height	Weight
Image View	match-parent	0dp	1
Text View	match-parent	wrap-content	0
Text View	match-parent	wrap-content	0

To create a linear layout in which each ChildView uses the same amount of space on the screen, set the

[android:layout\\_height](#)

of each View to "0dp" (for a vertical layout) or the

[android:layout\\_width](#)

of each View to "0dp" (for a horizontal layout).

Then set the

[android:layout\\_weight](#)

of each View to "1".

Weight = 0 (by default)

→ height & weight = wrap-content

# LinearLayout

- Examples

## HANGOUTS APP



Horizontal LinearLayout

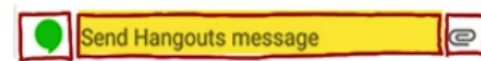
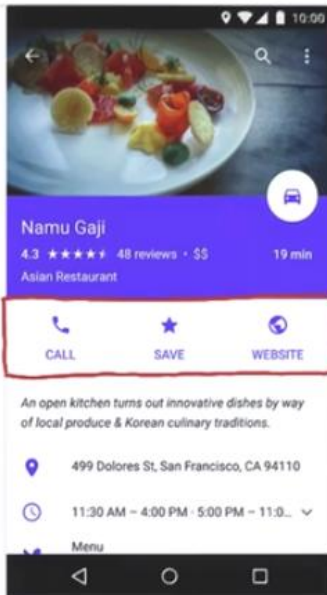


	Image View	EditText	Image View
Height	wrap_content	wrap_content	wrap_content
Width	wrap_content	0dp	wrap_content
Weight	0	1	0

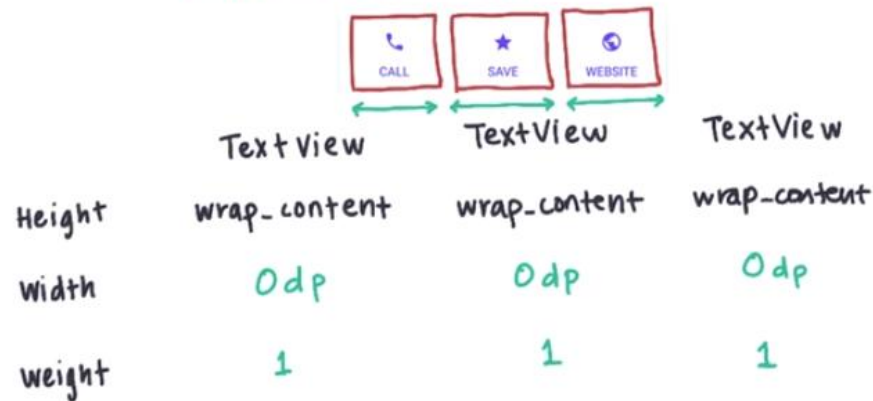
# LinearLayout

- Examples



## MAPS APP

### Horizontal LinearLayout




# LinearLayout

- Examples

EMAIL APP

Vertical Linear Layout



The screenshot shows an Android app interface for composing an email. It features a red header bar with a back arrow, the word 'Compose', and two icons. Below the header are four text input fields: 'From', 'To', 'Subject', and a large 'Compose email' body. A green double-headed arrow is placed to the right of the 'Compose email' field, indicating its height. The app is running on a device with a black navigation bar at the bottom.

	Width	Height	Weight
EditText	match-parent	wrap-content	0
EditText	match-parent	wrap-content	0
EditText	match-parent	wrap-content	0
EditText	match-parent	0dp	1

# ConstraintLayout

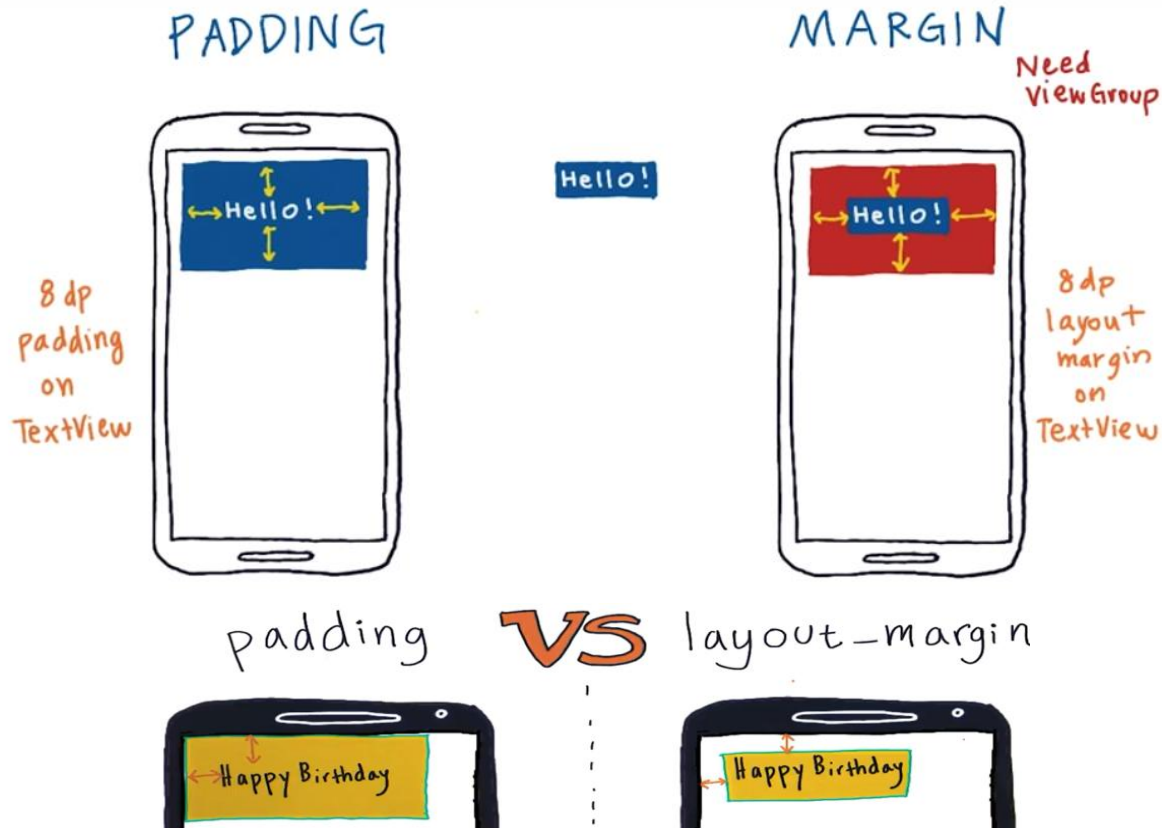
To build responsive UIs

- `app:layout_constraintLeft_toLeftOf="@+id/activity_main"`
- `app:layout_constraintTop_toTopOf="@+id/activity_main"`
- `app:layout_constraintRight_toRightOf="@+id/activity_main"`
- `app:layout_constraintBottom_toBottomOf="@+id/activity_main"`

<https://developer.android.com/studio/write/layout-editor.html>

<https://codelabs.developers.google.com/codelabs/constraint-layout/index.html?index=..%2F..%2Findex#5>

# PADDING VS. MARGIN

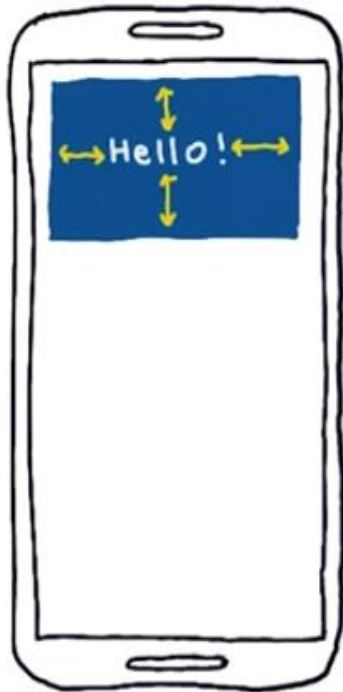


- Padding gets handled by the TextView – adds space within Views
- Margin gets handled by the ViewGroup – adds space around views

<https://material.io/guidelines/layout/metrics-keylines.html#metrics-keylines-baseline-grids>

# PADDING VS. MARGIN

- Padding



TextView attribute:

`android:padding = "8 dp"`

OR

`android:paddingLeft = "8 dp"`

`android:paddingRight = "8 dp"`

`android:paddingTop = "8 dp"`

`android:paddingBottom = "8 dp"`



# PADDING VS. MARGIN

- Margin

TextView attributes:

`android:layout_margin="8dp"`

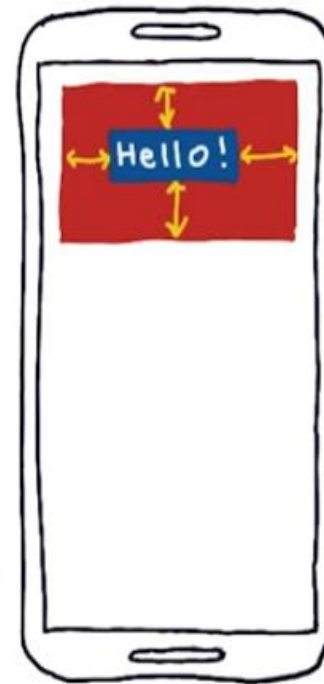
OR

`android:layout_marginLeft="8dp"`

`android:layout_marginRight="8dp"`

`android:layout_marginTop="8dp"`

`android:layout_marginBottom="8dp"`



Need  
View Group

# PROBLEM SOLVING

## Overlapping Views

```
<RelativeLayout>  
  <B>  
  <A>  
  <C>  
</RelativeLayout>
```



- Views can overlap each other
- The order of the Views tags determines the order that the Views get placed on the screen

# PROBLEM SOLVING

## DEBUGGING STEPS

1. Read the error message  
(copy the whole error message and paste that into a Google search)
2. Compare to working code samples  
(Common Android Views cheat sheet  
<https://drive.google.com/file/d/0B5XIkMkayHgRMVljUVIyZzNmQUU/view>)
3. Undo  
(command+z on Mac or ctrl+z on Windows to UNDO /  
command+shift+z or ctrl+shift+z to REDO)
4. Ask for help  
(upload a screenshot of your code on forums/chats  
<https://developer.android.com/studio/debug/am-screenshot.html>)