Agenda 455/555 9.9.18 Mon.

* Chapter read: 21-22, 26(pgs. 265-269)
* Points to ponder

-Adding different AVD's

-Troubleshooting

\*Build / Rebuild / Resync gradle

Getting familiar with your Emulator!

Dismissing (hiding) your onboard keyboard

Control + arrow keys @ runtime to reorient screen, etc

Snapshotting toasts w long lengths-- set snipping tool timer, or with Skitch (for the Mac)

* General layouts

**Relative vs Linear**

[**RelativeLayout**](https://developer.android.com/guide/topics/ui/layout/relative) : Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

[**LinearLayout**](https://developer.android.com/guide/topics/ui/layout/relative) : A layout that organizes its children into a single horizontal or vertical **row**.

[**Constraint**](https://stackoverflow.com/questions/37321448/differences-between-constraintlayout-and-relativelayout)**:** A **ConstraintLayout** is a **ViewGroup** which allows you to position and size widgets in a flexible way!

[**https://developer.android.com/reference/android/support/constraint/ConstraintLayout**](https://developer.android.com/reference/android/support/constraint/ConstraintLayout)

Dimensions / Measurements – for screen sizing, fonts, element positioning, etc

Dp measurements – Dp’s vs. Pixels

<https://developer.android.com/guide/practices/screens_support.html>

<https://material.io/guidelines/layout/units-measurements.html#units-measurements-density-independent-pixels-dp>

A dimension value defined in XML. ( 4 ) Basic- A dimension is specified with a number followed by a unit of measure. For example: 10px, 2in, 5sp.

**android:layout\_width="52dp"  
android:layout\_height="46dp"**

The following units of measure are supported by Android:

Dp or dip

**Density-independent Pixels** - An *abstract* unit that is based on the physical density of the screen. These units are relative to a 160 dpi (dots per inch – known as the baseline) screen, on which 1dp is roughly equal to 1px.

When running on a higher density screen, the number of pixels used to draw 1dp is scaled up by a factor appropriate for the screen's dpi. Likewise, when on a lower density screen, the number of pixels used for 1dp is scaled down.

The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. Using **dp units** (instead of px units) is a simple solution to making the **view dimensions** in your layout resize properly for different screen densities. In other words, it provides **consistency** for the real-world sizes of your UI elements across different devices.

sp

**Scale-independent Pixels** - This is like the dp unit, but it is also scaled by the user's **font** size preference. It is recommend you use this unit when specifying font sizes, so they will be adjusted for both the screen density and the user's preference.

pt

**Points** - 1/72 of an inch based on the physical size of the screen, assuming a 72dpi density screen.

px

**Pixels** - Corresponds to actual pixels on the screen. This unit of measure is not recommended because the actual representation can vary across devices; each devices may have a different number of pixels per inch and may have more or fewer total pixels available on the screen.

Other

mm

**Millimeters** - Based on the physical size of the screen.

in

I**nches** - Based on the physical size of the screen.

Ex.

XML file saved at res/values/dimens.xml:

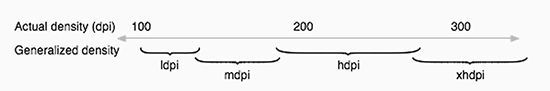
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <dimen name="textview\_height">25dp</dimen>  
    <dimen name="textview\_width">150dp</dimen>  
    <dimen name="ball\_radius">30dp</dimen>  
    <dimen name="font\_size">16sp</dimen>  
</resources>

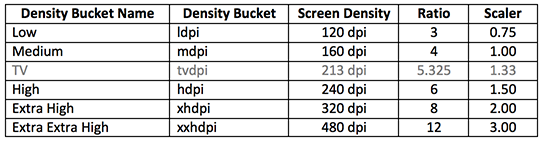
Screen Densities

Screen density is a ratio of resolution and display size, and both factors contribute to the density.

density equals

* .75 on ldpi (120 dpi)
* 1.0 on mdpi (160 dpi; baseline)
* 1.5 on hdpi (240 dpi)
* 2.0 on xhdpi (320 dpi)
* 3.0 on xxhdpi (480 dpi)
* 4.0 on xxxhdpi (640 dpi)





<https://material.io/tools/devices/>

Q. DP calculation

To calculate dp’s needed for a screen layouts we can use the formula

**dp = (width in pixels \* 160) / density**. What would be the resulting icon size

for a 200x200 px icon given a screen density bucket size of 320?



**200x200px Image at xhdpi (320dpi)**

**Ans . 100dp**

<https://pixplicity.com/dp-px-converter>

**Defining UI Element Bounds (wrap vs. parent)**

When defining width and height attributes on a user interface element, there are special options available, as well as the dimension units.

**wrap\_content** - This will “wrap” the bounds of the element to be just large enough to contain its content (images, text, etc), children elements contained within it, plus padding. Essentially, this set the element be the size of its largest content, and will not adjust the size of the element.

**match\_parent** (fill\_parent deprecated in API 8) - This will “match” the bounds of the element to its parent element, using the maximum allowed space, minus padding. Essentially, this lets the element be the max size its parent allows, and will adjust the size of the element if needed.

* Lab 2 – continues with lab 1 and introduces a Checkbox, SeekBar, listview with an ArrayAdapter and a Viewstub.

View controls

**ViewStub** - A ViewStub is an invisible, zero-sized View that can be used to lazily *inflate*

layout resources at runtime.

<https://developer.android.com/reference/android/view/ViewStub.html>

**Seekbar** - A SeekBar is an extension of ProgressBar that adds a draggable thumb.

<https://developer.android.com/reference/android/widget/SeekBar.html>

**Listview** -  is a view group that displays a list of scrollable items.

<https://developer.android.com/guide/topics/ui/layout/listview.html>

**Arrayadapter** widget - A concrete BaseAdapter that is backed by an array of arbitrary objects.

<https://developer.android.com/reference/android/widget/ArrayAdapter.html>

Simple\_list\_item - It tells the **listview** what layout to use for the individual rows.

<https://developer.android.com/reference/android/R.layout.html>

<http://stackoverflow.com/questions/6079344/what-is-android-r-layout-simple-list-item-1>

Event Listeners/Handling,

SeekBar.[OnSeekBarChangeListener](https://developer.android.com/reference/android/widget/SeekBar.OnSeekBarChangeListener) is a ***callback*** that notifies clients when the progress level has changed for the seek bar control by touch gestures, arrow keys/trackballs, etc. and has three possible abstract methods that need to be overriden for implementation. Two of the methods are onStartTrackingTouch and onProgressChanged. Can you name the third needed overriden method to implement?

abstract void **onProgressChanged(**SeekBar seekBar, int progress, boolean fromUser)

Need: Notification that the progress level has changed.

abstract void **onStartTrackingTouch(**SeekBar seekBar)

Need: Notification that the user has started a touch gesture.

abstract void **onStopTrackingTouch**(SeekBar seekBar)

Need: Notification that the user has finished a touch gesture.