# Building@Speed of Thought

## SysCore DeSign

Siamak (Ash) Ashrafi
CTO - ZoeWave.com
@TheGoodTomChi

*with inspiration from* →

Tom Chi

APRIL 08+09
DROIDCON BOS19

# Description

Breaking away from the traditional steps of a design sprint (http://www.gv.com/sprint) we blur them together to further reduce the development cycle in what we call "**Building @ Speed of Thought**". Over the years my group has been perfecting how to build world-class Android Apps at the speed of thought first for hackathons, then products and now for design sprints. In this session, we will detail the "Speed of Thought" methodology, steps, tools and components involved. We start by utilizing a **design sprint** to explore the power of mobile and how it fits into the market/problem we are trying to solve. Then focusing on the crucial step in a design sprint we build high fidelity prototypes in rapid succession utilizing the **JetPack Android** software components (UI, Behavior, Foundation, Architecture) to take full advantage of the Android platform. We do this by leveraging the design features of **Android Studio** (material components/theme editor, constraint library, motion editor) along with open source/industry resources to build an intuitive and beautiful UI. Once the UI framework is done we write the **Kotlin** application logic utilizing the Android Architecture Components with data binding connected to Firebase for the synchronized backend. With our application finished we execute our strategy for marketing, distribution, and monetization guided by Data-Driven Ethnography (**DDE**). At the end of the session, the audience will have a good idea of how in rapid secession all these tools and components come together to **produce magic**.

# Design of Mobile

- Code: **Design** Patterns To do the **thing right**

Starting with the basics of writing clean code (SOLID, AAC, TDD & DI with Kotlin)

- Interface: **Design** UX/UI To do the **right thing**

Applying Material Design U&I with the latest research we build intuitive UX

- Product: **Design** Thinking The **right** thing to **do**

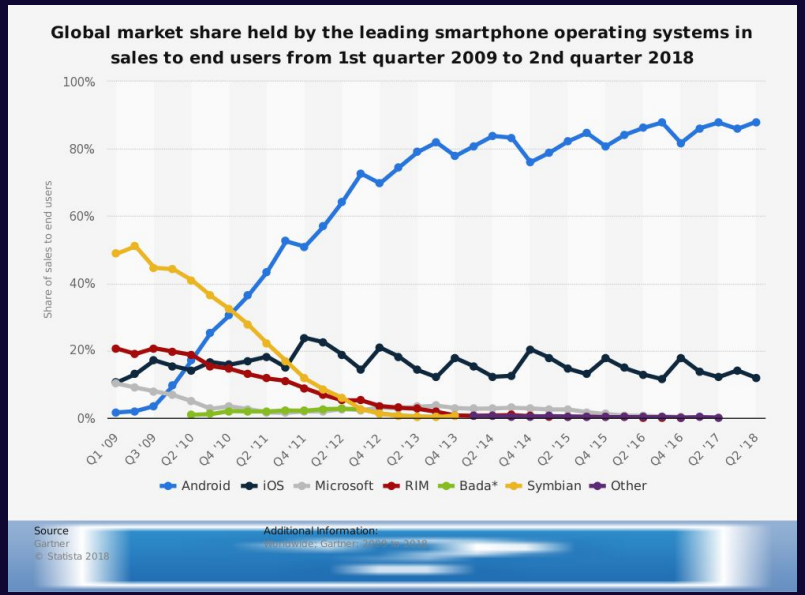
Design Thinking uses iterative data driven ethnography to build empathy

# Who’s Mobile ???

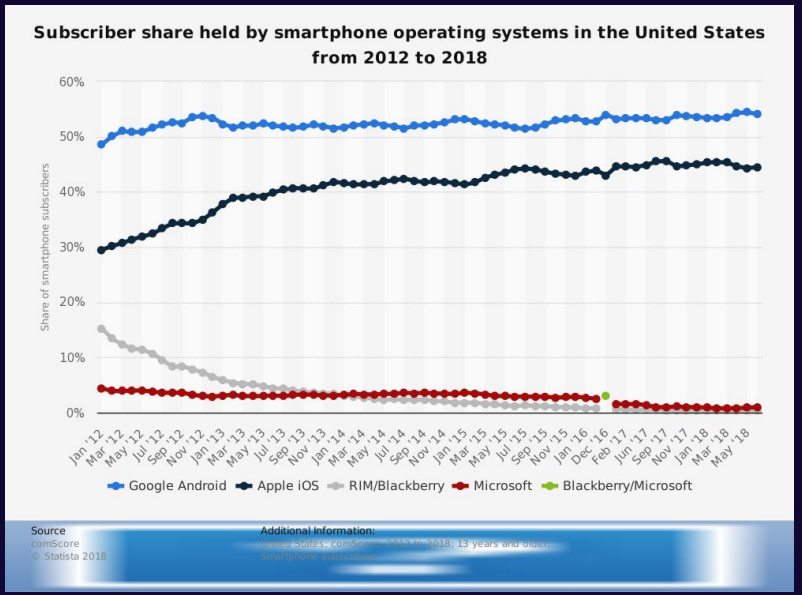
# Connecting Android & iOS

# iOS vs Android

## Android vs iOS in World




## Android vs iOS in US



# Android & iOS Innovation

Great minds think alike - or just copy each other?

# Android & iOS


androidcentral


AT&T
9:42 AM
Text
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Clock
Calculator
Notes
Settings

# Kotlin & Swift

http://nilhcem.com/swift-is-like-kotlin

## Functions & Protocols

Kotlin

```kotlin
fun makeIncrementer(): (Int) -> Int {
    val addOne = fun(number: Int): Int {
        return 1 + number
    }
    return addOne
}
val increment = makeIncrementer()
increment(7)

// makeIncrementer can also be written in a shorter way:
fun makeIncrementer() = fun(number: Int) = 1 + number
```

Swift

```swift
func makeIncrementer() -> (Int -> Int) {
    func addOne(number: Int) -> Int {
        return 1 + number
    }
    return addOne
}
let increment = makeIncrementer()
increment(7)
```

Kotlin

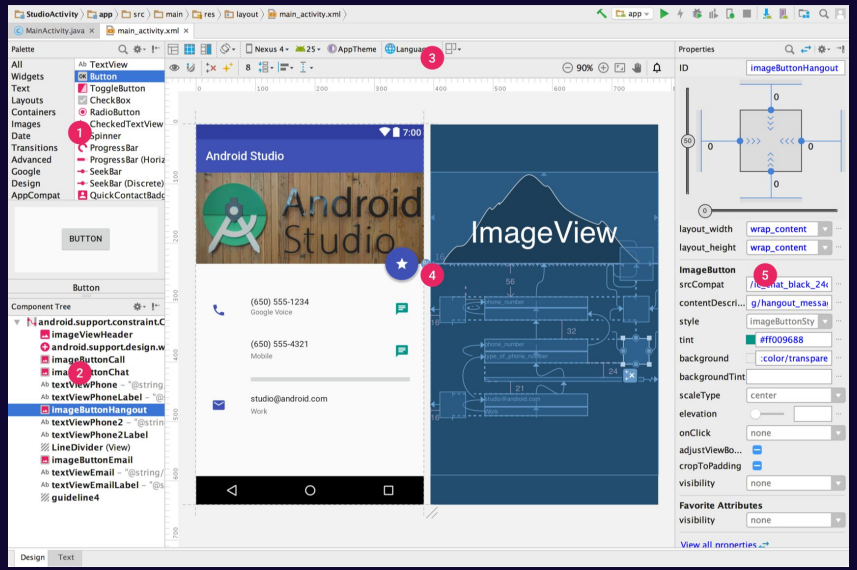
```kotlin
interface Nameable {
    fun name(): String
}

fun f<T: Nameable>(x: T) {
    println("Name is " + x.name())
}
```

Swift

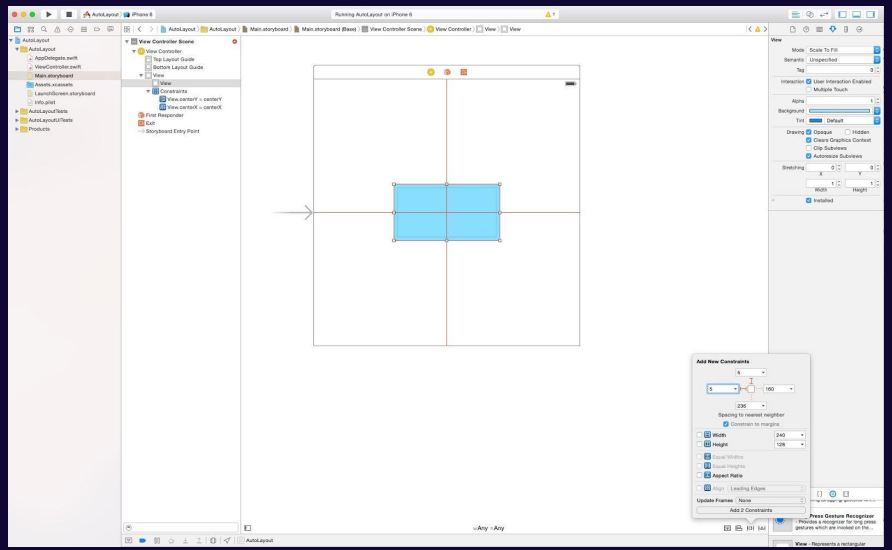
```swift
protocol Nameable {
    func name() -> String
}

func f<T: Nameable>(x: T) {
    print("Name is " + x.name())
}
```

# Android Studio & XCode

Constraint System with LiveData



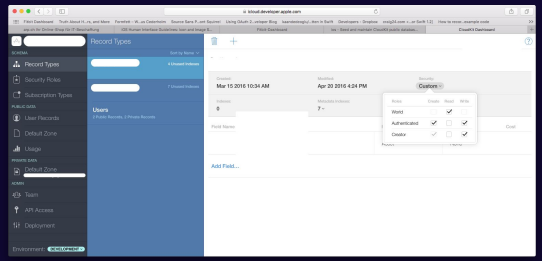

Constraint System with NSFetchedResultsController

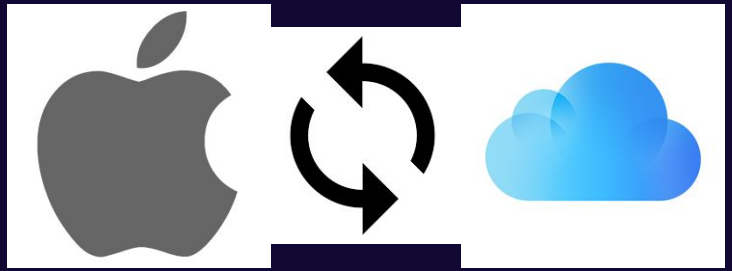


# Cloud DB & Messaging Backend

Apple Notification Center

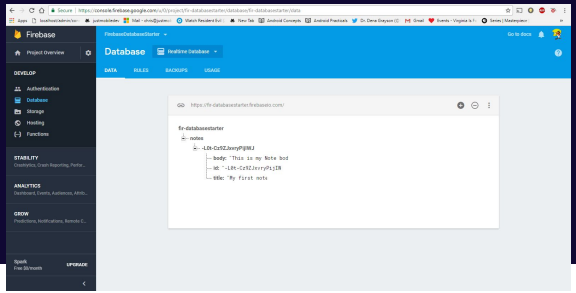
iCloud / Firebase realtime DB

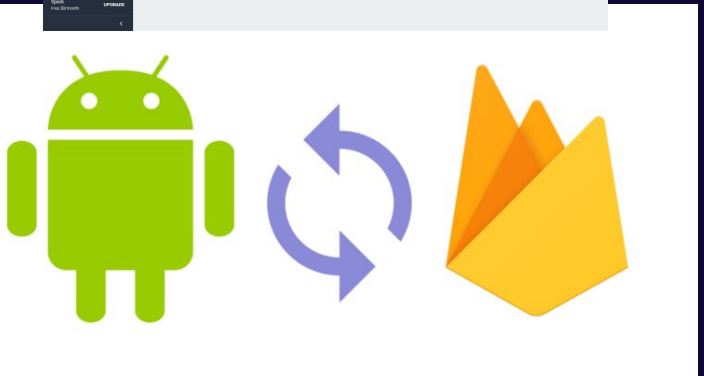




Firebase Cloud Message (GCM)

Firebase realtime DB



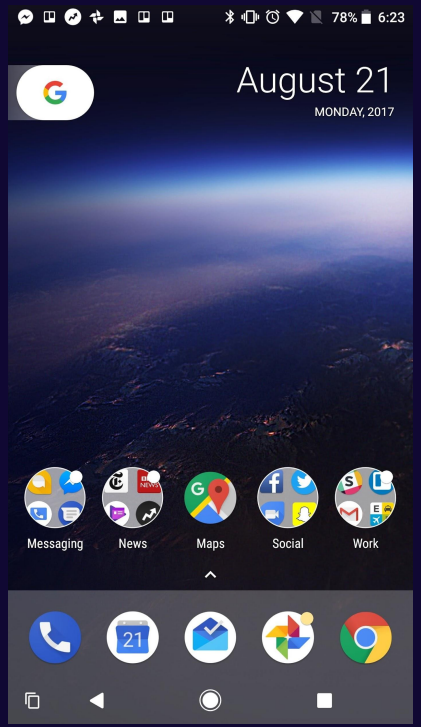


# Material Design & Apple HIG

Icon Folders

Bottom Tab

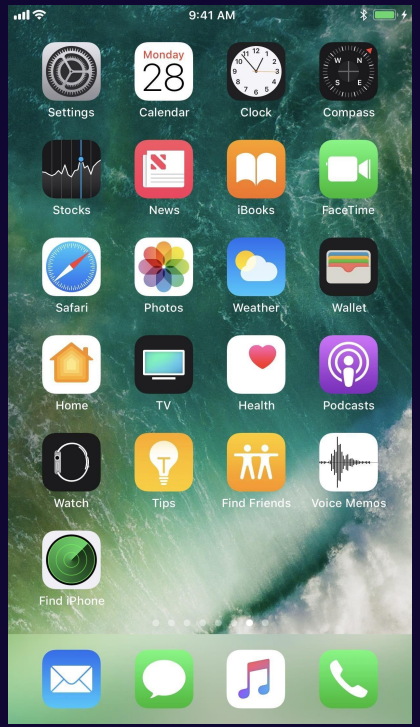
Swipe Nav




Icon Folder

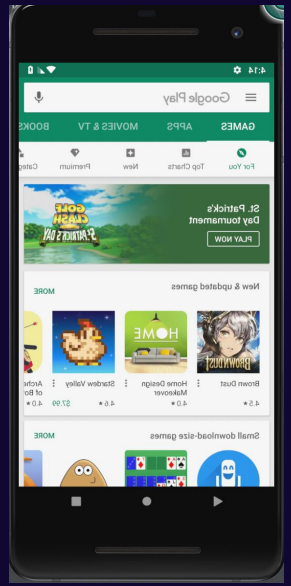
Bottom Tab

Swipe Nav

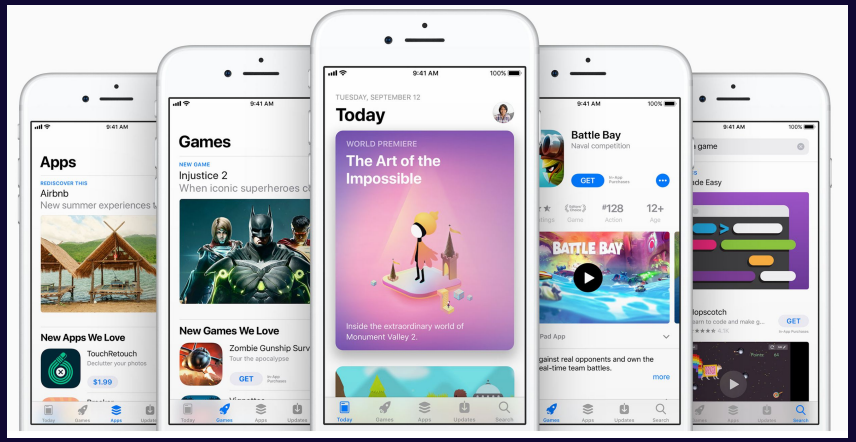


# Google Play Store & iOS App Store

New Clean Design



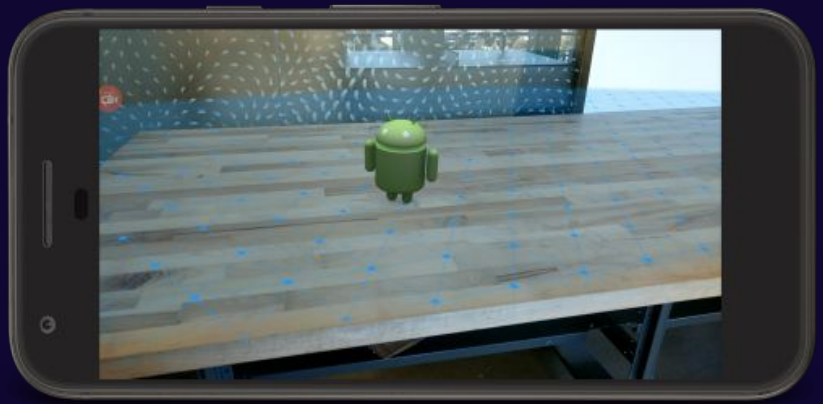
New Clean Design



# The Power of Mobile!

# ARCore 2 & ARKit 2

Built using Unity or Unreal Engine



Built using Unity or Unreal Engine

# ARCore with Sceneform

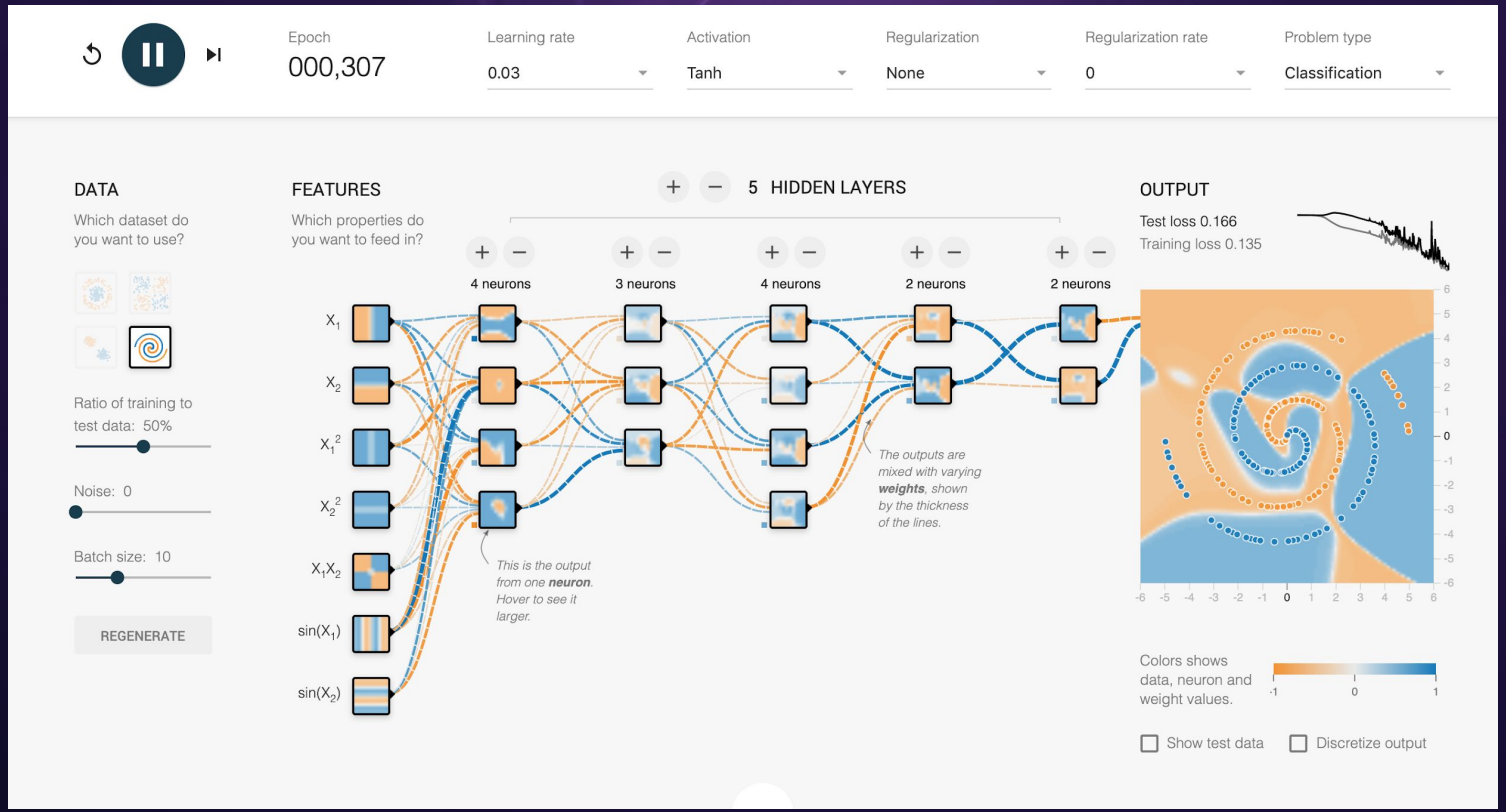
```
arFragment = (ArFragment)findFragmentById(R.id.x);


ModelRenderable.builder()
        .setSource(this, R.raw.andy)
        .build()


arFragment.setOnTapArPlaneListener()
anchorNode.setParent(arFragment.getArSceneView().getScene
```

# Artificial Intelligence / Machine Learning

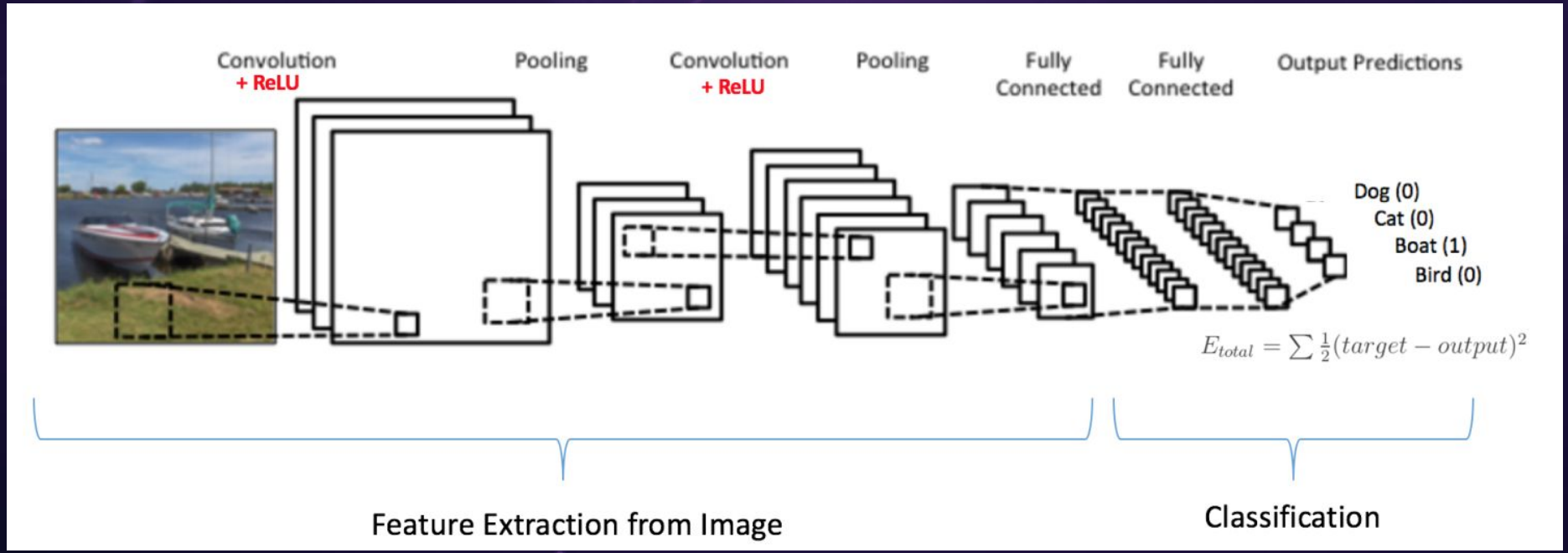
TensorFlow Lite

CoreML

# TensorFlow Playground

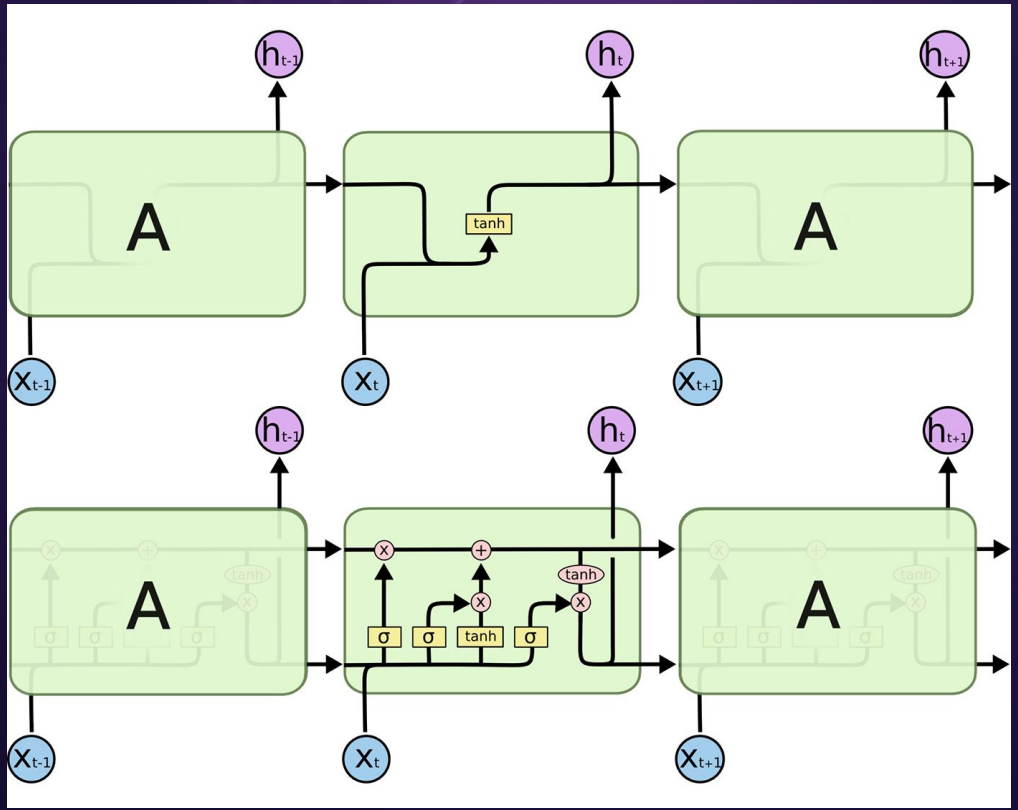


# Convolutional Neural Networks :-( *CNN is NOT fake news!* :-)




https://cloud.google.com/blog/products/gcp/learn-tensorflow-and-deep-learning-without-a-phd

# Building with TensorFlow

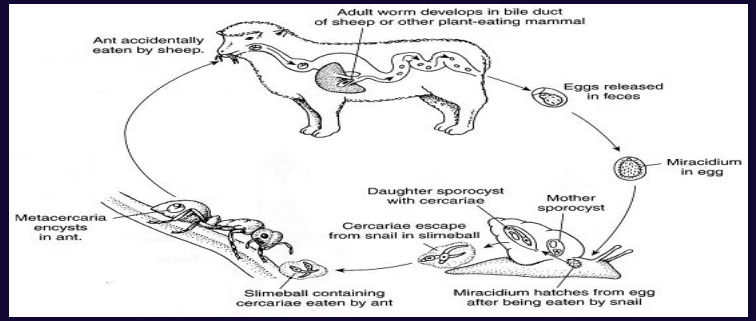


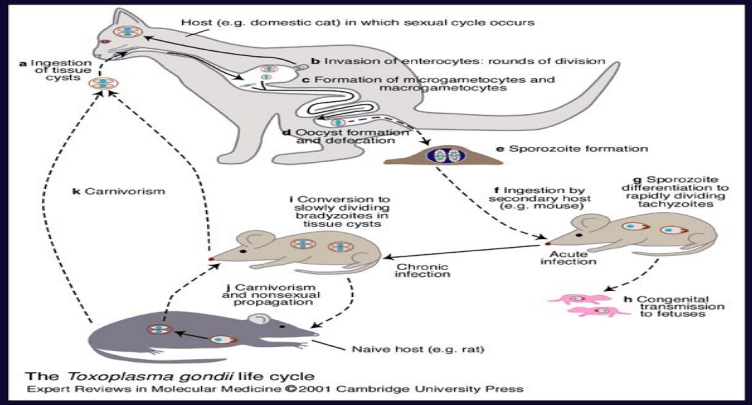
# Sensors / Connectivity / Personal Info*

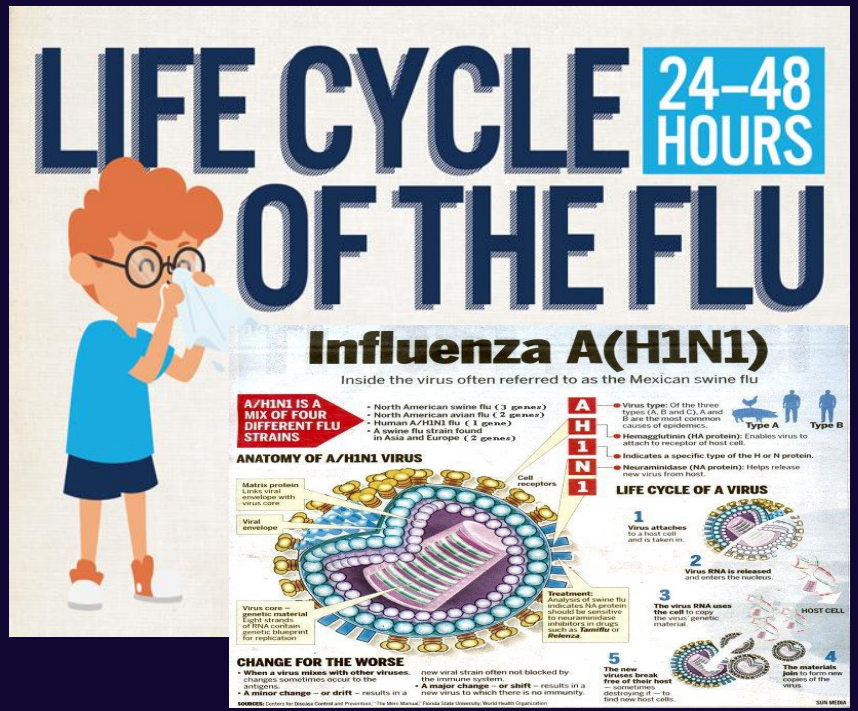
Android platform supports three broad categories of sensors:

- Motion sensors - These sensors measure acceleration forces and rotational forces along three axes.
- Environmental sensors - These sensors measure various environmental parameters.
- Position sensors - These sensors measure the physical position of a device.

# The Power Of Mobile in Mind Control



# Building @ Speed of Thought

Extreme DeSign Sprints

**Product Design**
The **right** thing to **do**

**Functional & Practical**

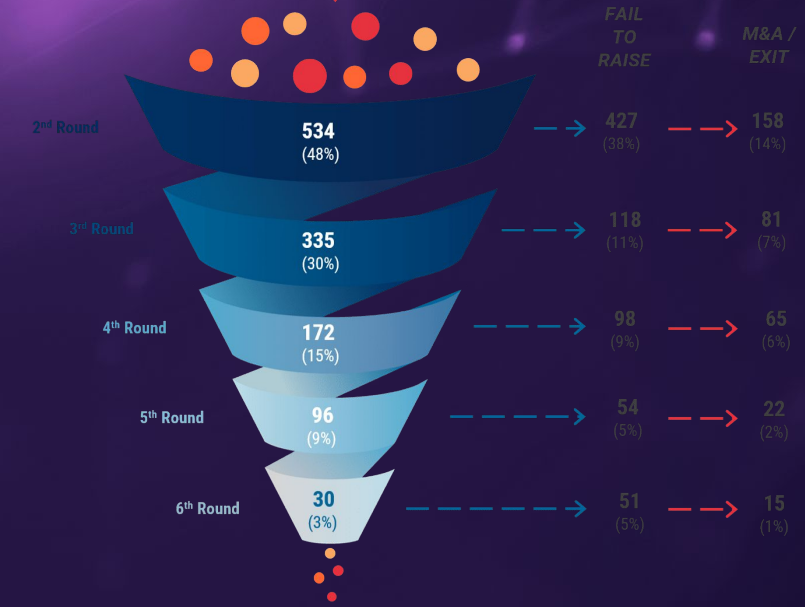
# What's the Problem

# TOP 7 DEAD CONSUMER HARDWARE STARTUPS BY AMOUNT RAISED

| | | |
|---|---|---|
| JAWBONE | $930M Dead | Wearable |
| NJOY E-CIGS \| VAPING | $181M Asset Sale | E-Cig<br>*NJOY assets sold to Homewood Capital* |
| JUICERO | $100M Dead | Juicer |
| fuhu | $75M Asset Sale | Tablet<br>*FUHU (creator of nabi tablet) assets sold to Mattel* |
| pebble | $59M Asset Sale | Wearable<br>*Pebble assets sold to Fitbit* |
| zeebo | $54M Dead | Game Console |
| hello | $53M Dead | Sleep Tracker Sensor |

** $M shown of amount raised before Dead/Asset Sale*

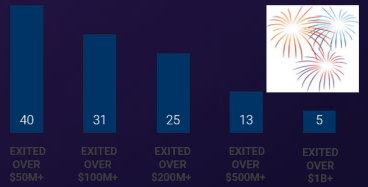
## 1119 US Seed Tech Companies




67% DEAD / SELF-SUSTAINING

1% (12 companies) REACHED UNICORN STATUS INCLUDING . . .

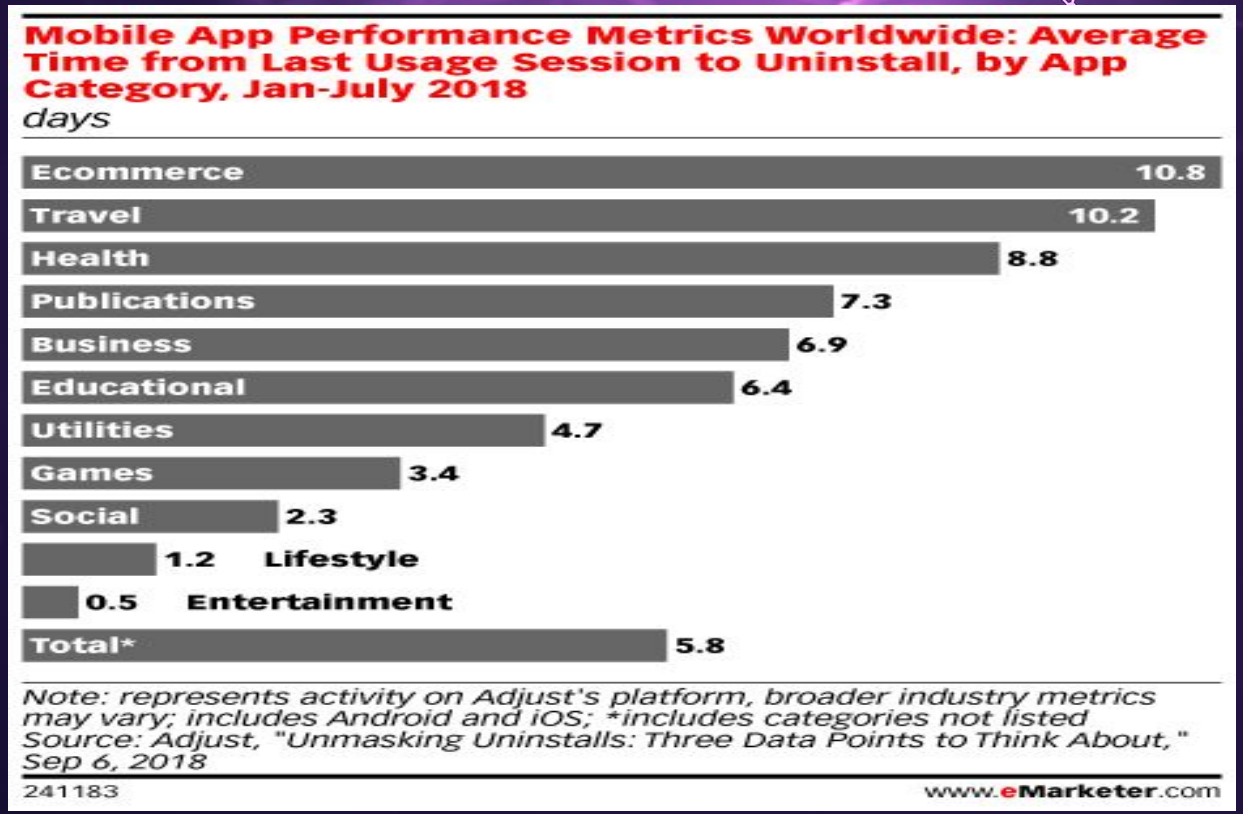
stripe docker




Building it wrong.

Note: All numbers based on cohort of companies that raised Seed in 2008, 2009 or 2010 and disclosed valuations only.

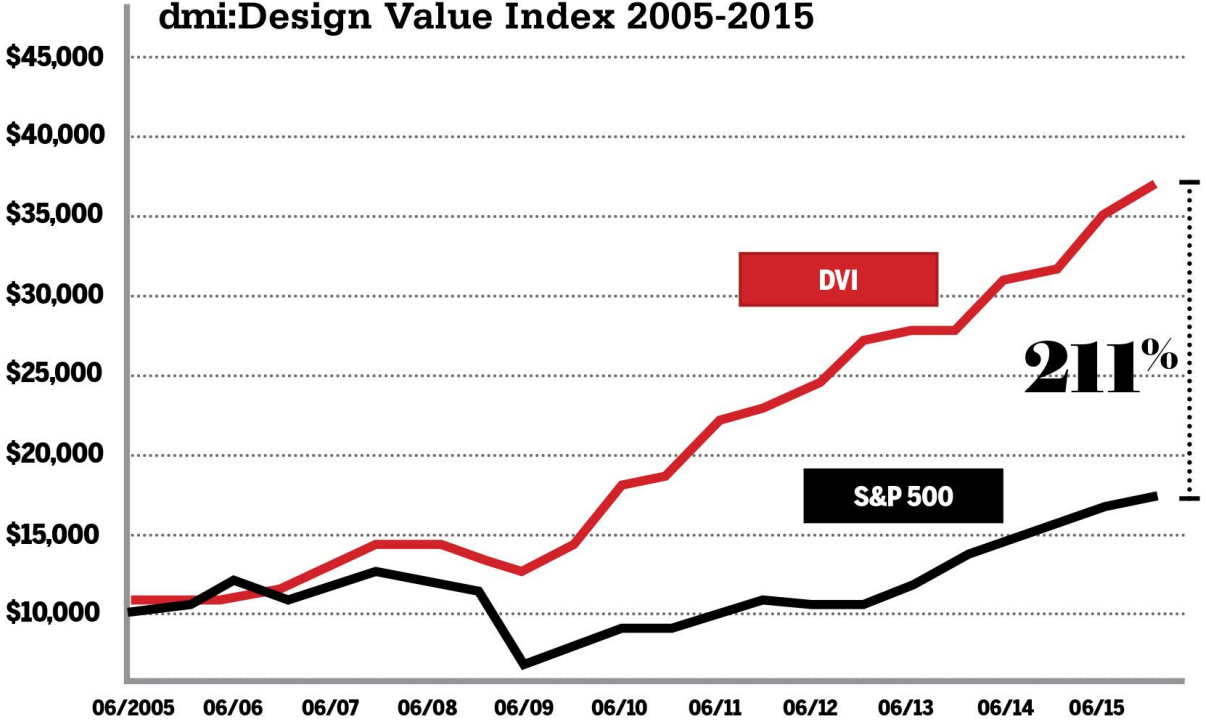
CBINSIGHTS




Todays mobile apps are not very good

**DESIGN-CENTRIC COMPANIES:**

APPLE
COCA-COLA
FORD
HERMAN-MILLER
IBM
INTUIT
NIKE
PROCTER & GAMBLE
SAP
STARBUCKS
STARWOOD
STANLEY BLACK & DECKER
STEELCASE
TARGET
WALT DISNEY
WHIRLPOOL






Facebook, Google, and Amazon have collectively grown art and design headcount by 65% in the past year

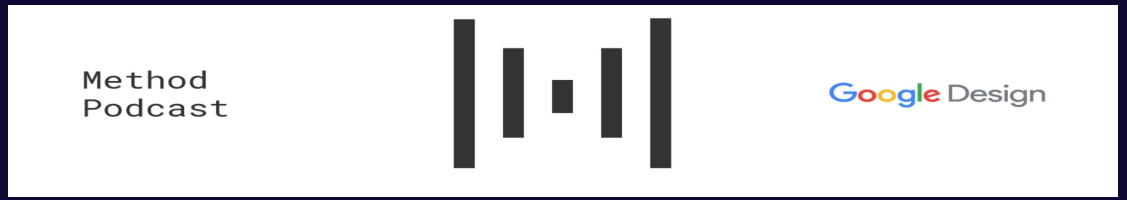
# De (X) Sign - Understand / Signal

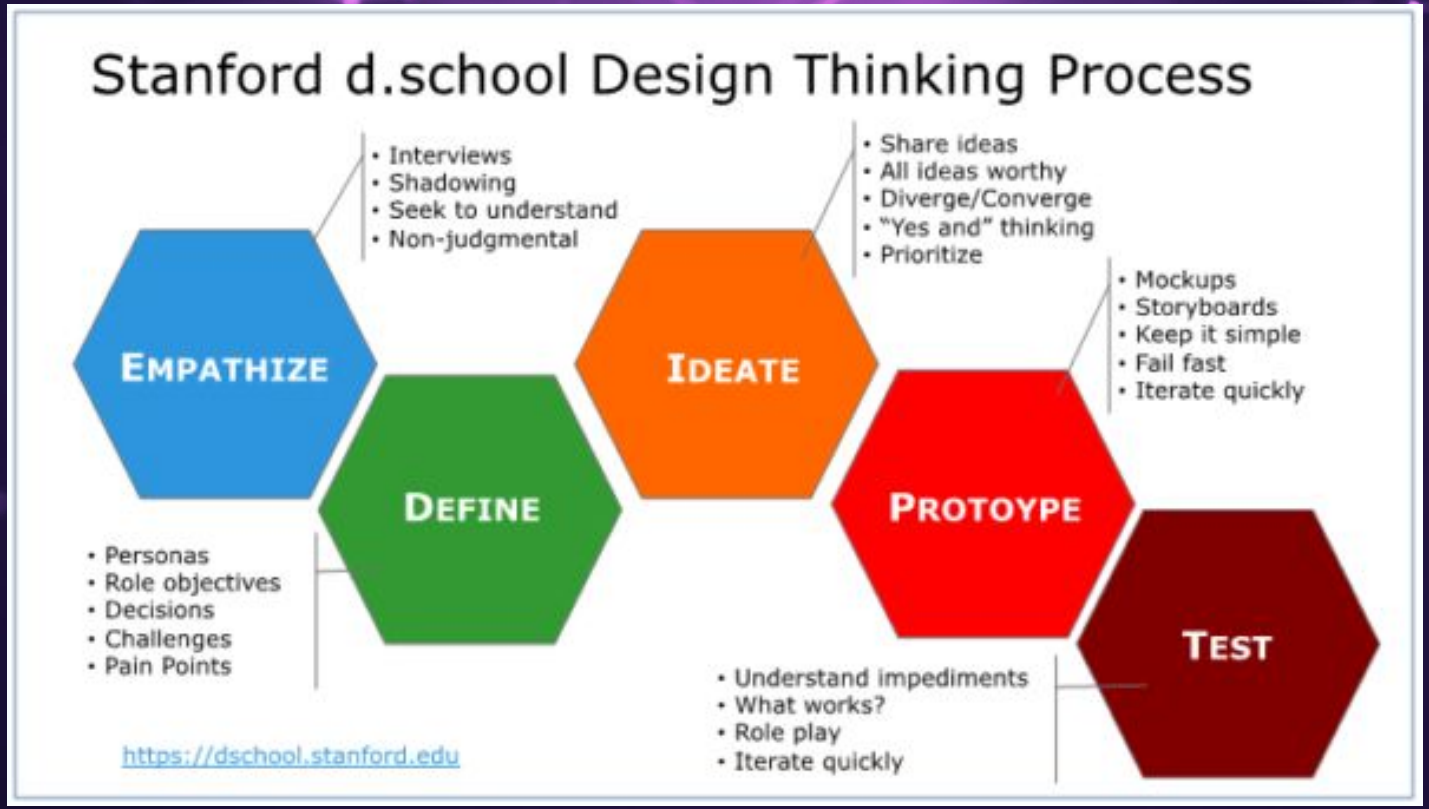
# Google DeSign

https://design.google/

Design Is

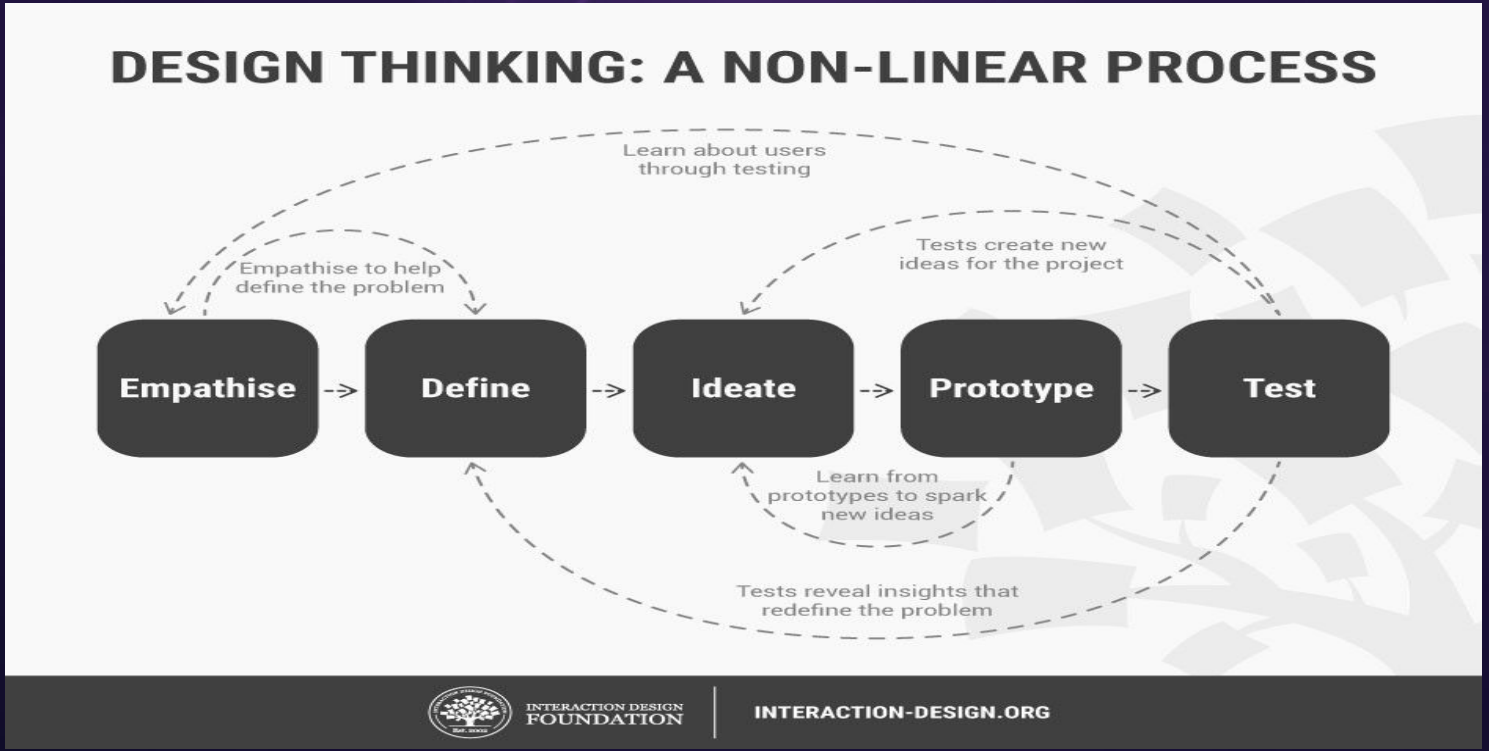







DeSign Thinking Started it All
https://www.ideou.com/pages/design-thinking

# Many Steps till User

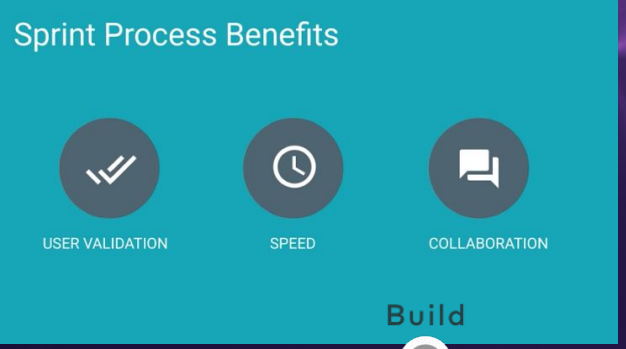


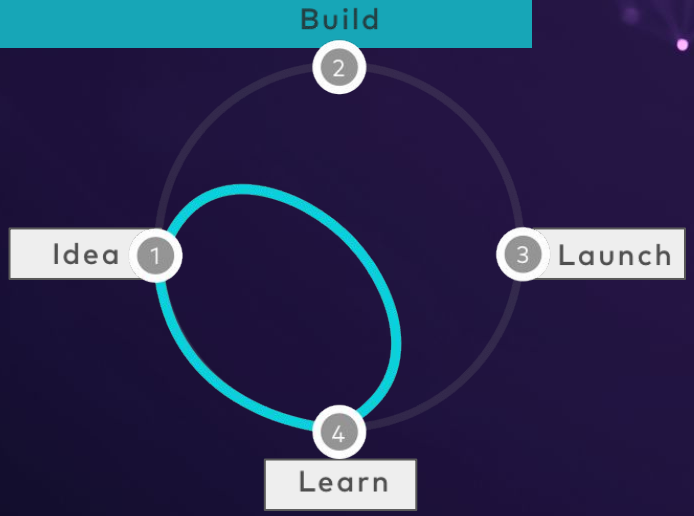
# DeSign Sprints

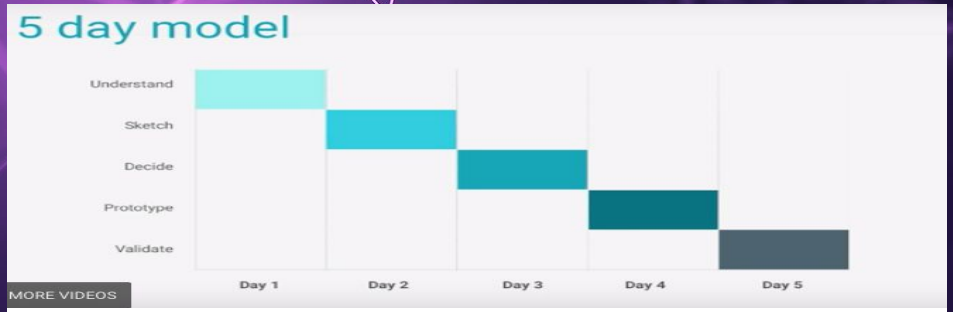
# DeSign Sprint

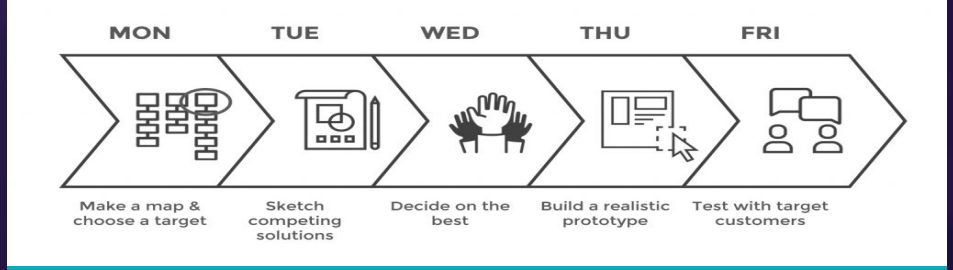
Design sprints enable you to:

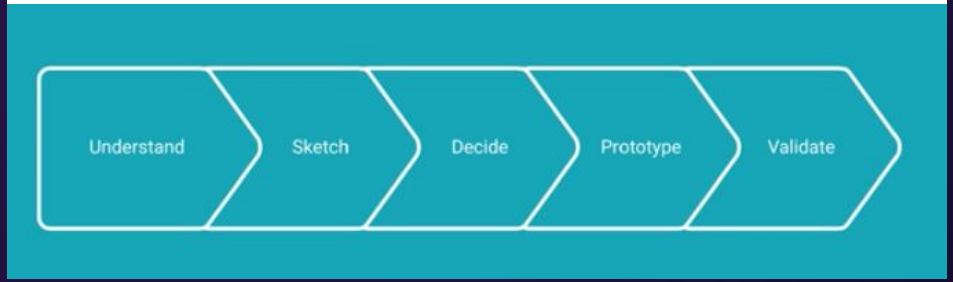
- Clarify the problem at hand, and identify the needs of potential users
- Explore solutions through brainstorming and sketching exercises
- Distill your ideas into one or two solutions that you can test
- Prototype your solution and bring it to life
- Test the prototype with people who would use it



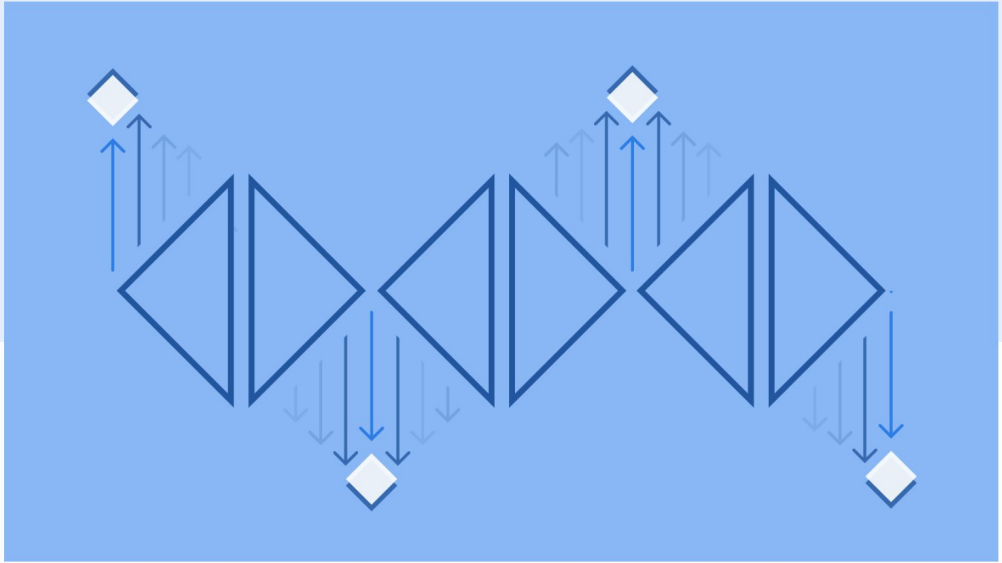




DeSign Sprint Process
https://www.gv.com/sprint/

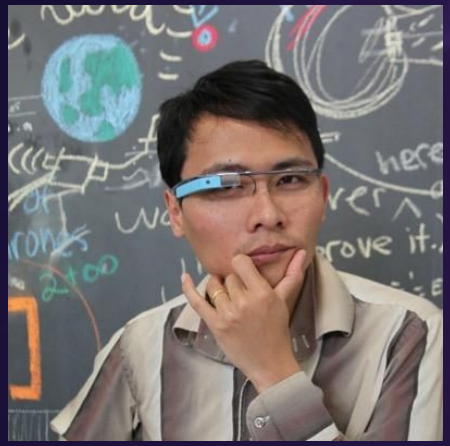
# Library

Find and discover the tools, resources, and templates that work best for your team's Design Sprint.



Methodology
Methods Overview

https://designsprintkit.withgoogle.com/resources

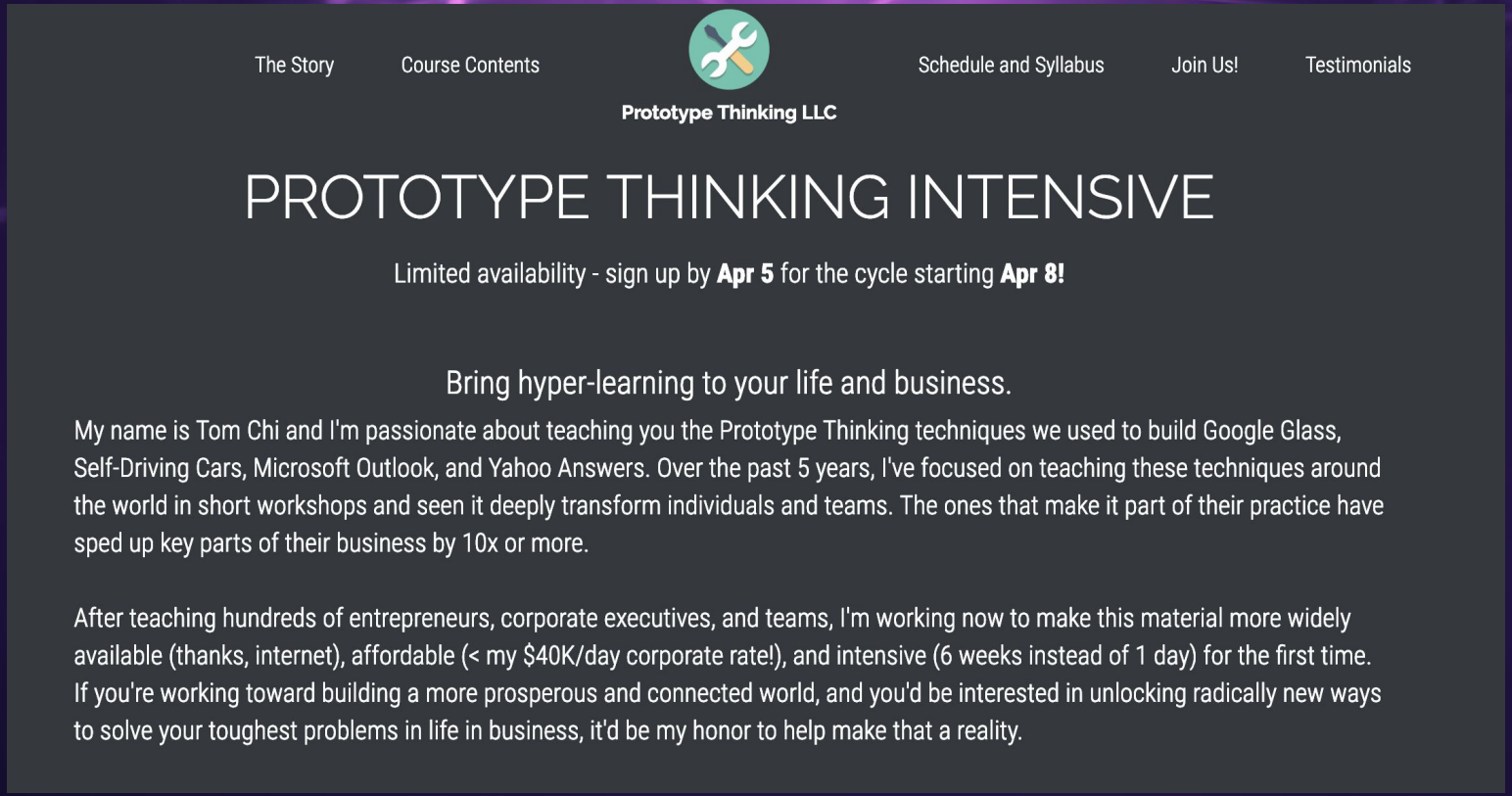



[Tom Chi ](#) - Home & Away Teams

# Tools for Rapid Development


- Tools as close as possible
- Cross functional
- Build to the solution
- Do not let ego keep you from success

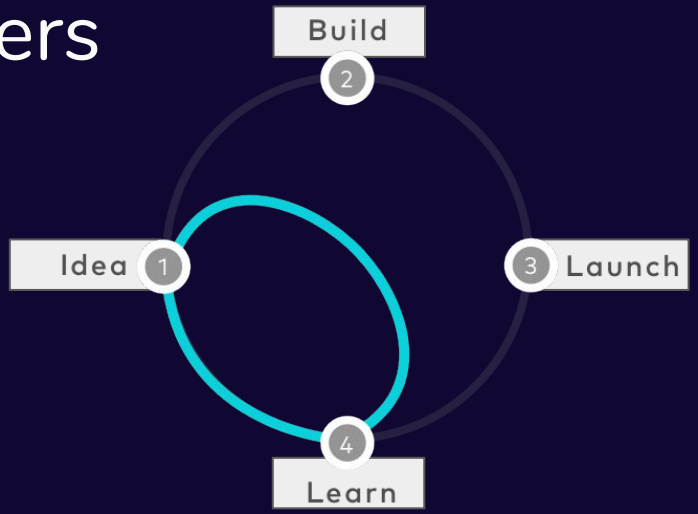
Sprint Conference: https://www.youtube.com/watch?v=z_elgzL9sns



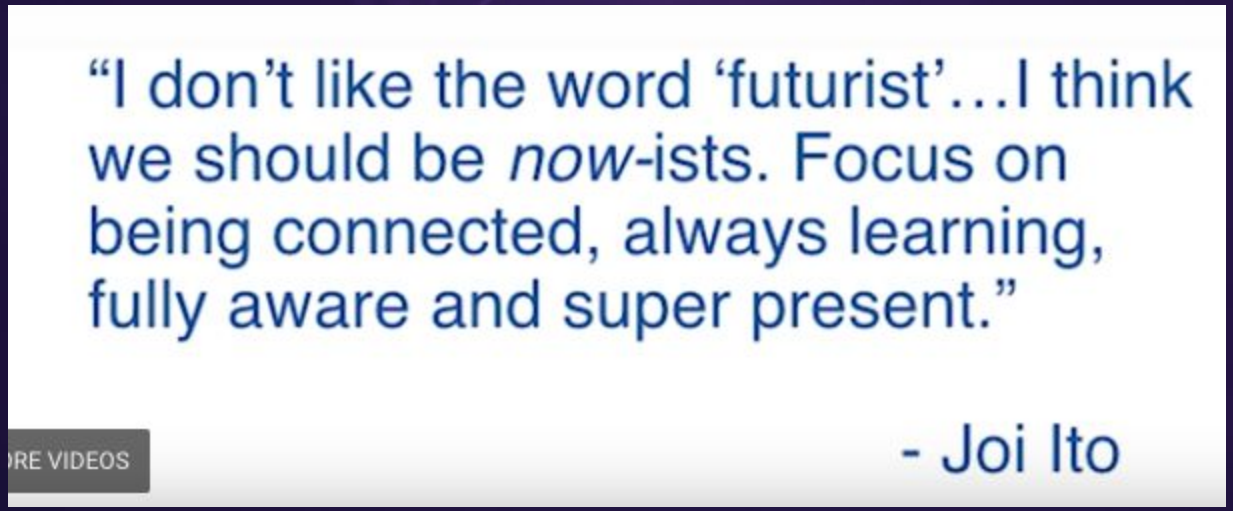

https://www.prototypethinking.live/

## Shorten the USER feedback loop

- Proct <-> UX <-> Dev -> X -> Users
- Design <-> Build <-> Users
- Build <-> Users

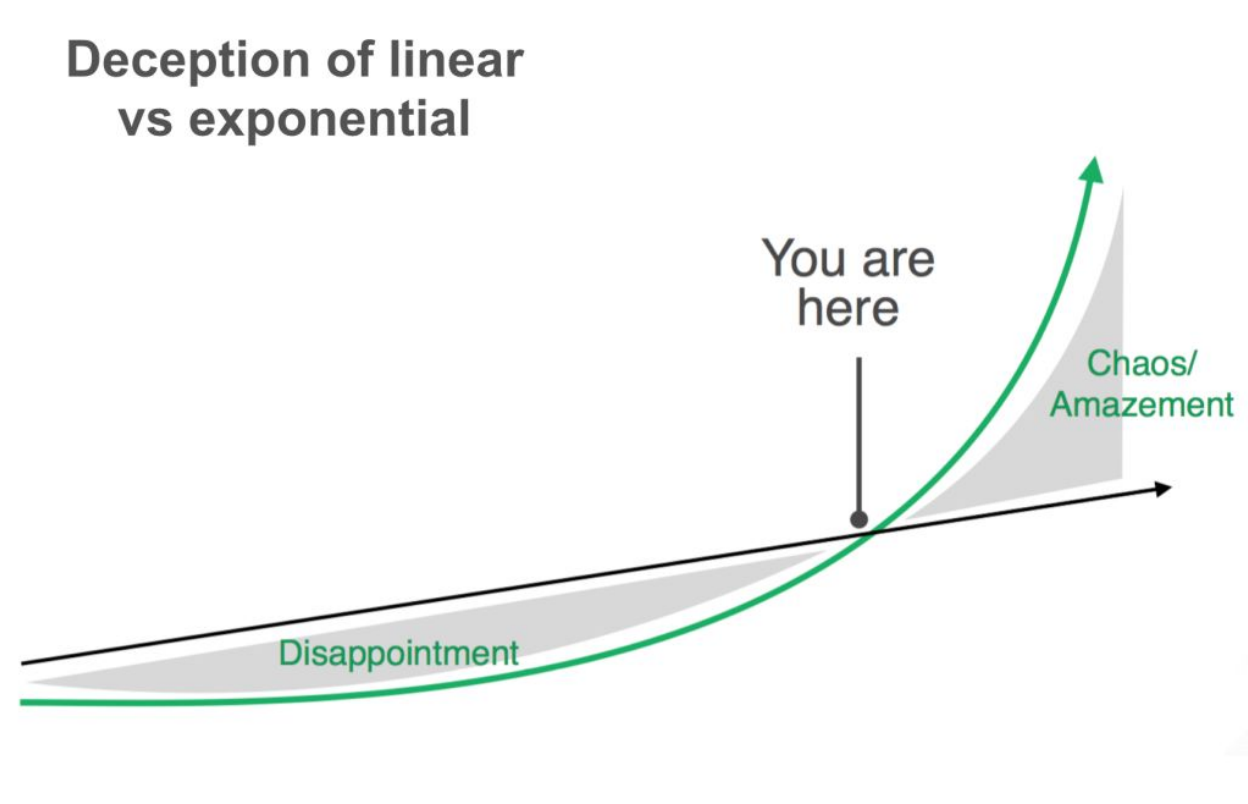


# Building in real-time




The best way to predict the future is to create it -Alan Kay

# The Descriptive Curve of Technology



# 45 Years and 20 Min

# Burning Clay

Art & Fear

by David Bayles and Ted Orland

1. graded solely on the quantity of work they produced
2. graded solely on its quality.




User Feedback


DeSign Products.

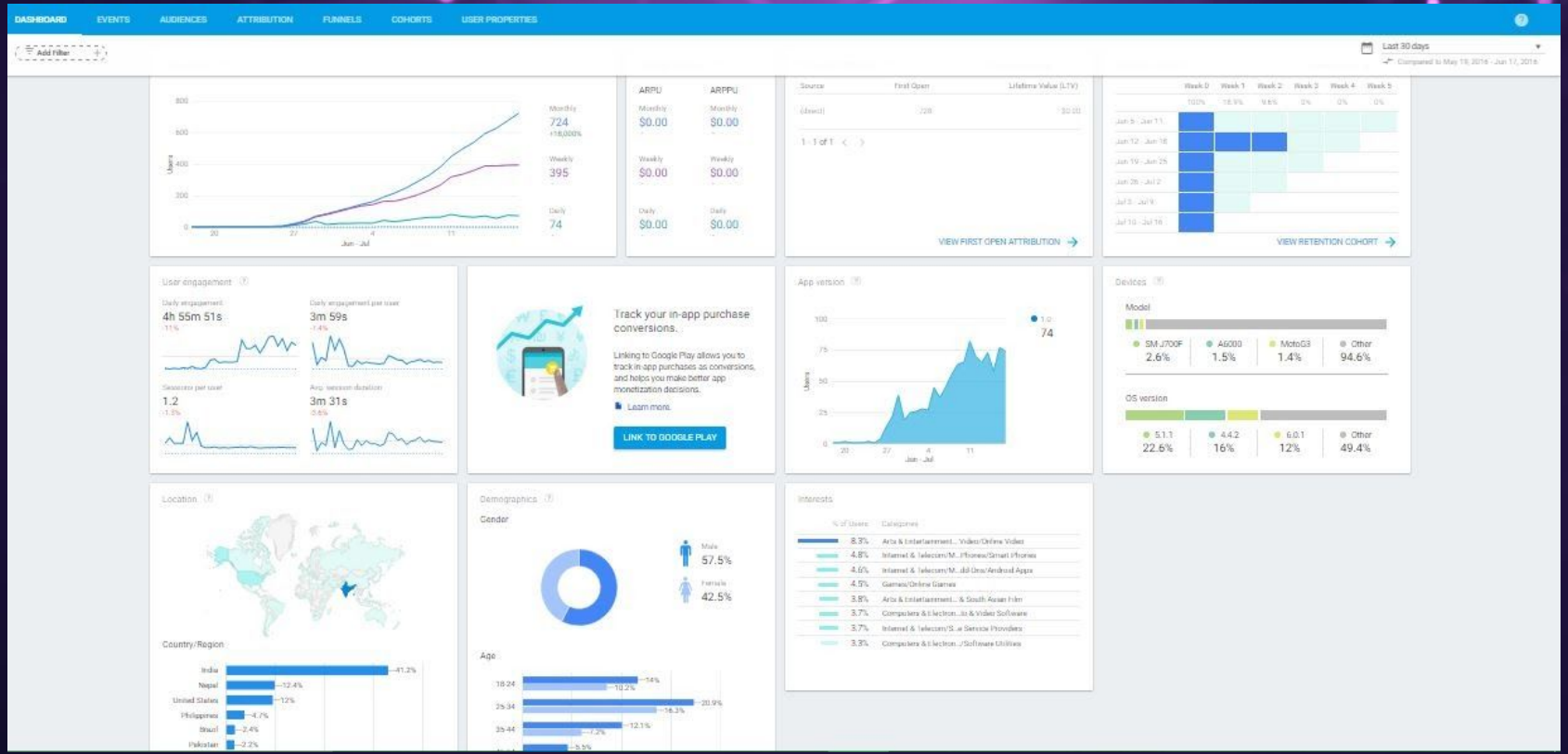
# Data Driven Ethnography

The user journey is in the data.

Valentine's Day they changed the App Icon to pink: installs dropped by 20%

Respondents Lie:

- Respondents want to appear better than they are.
- Respondents give socially desirable answers.
- They don’t want to answer questions about sensitive behavior.




Data Driven Ethnography -- Do not listen to your users.

# ThinkWithGoogle.com

- Google Marketing Research & Digital Trends
- What you need to know about mobile app users
- Principles of Mobile App Design: Engage Users and Drive Conversions.
- The Customer Experience is Written in Data
- Secrets to App Success on Google Play (A / B Testing)
- https://developers.google.com/analytics/

*In God we trust … Everyone else bring DATA!

# UI Design

To do the right thing

Intuitive and beautiful

# The Goal


Material DeSign Awards


Apple DeSign Award

# UX/UI Design*

Visit the material design & developer design

Design is never done - Material Design's new suite of tools and guidelines

Designers Cheat Sheet - Android Cheatsheet for Graphic Designers

Chet & Romain: U & I - Information and techniques for making better UIs

*Illusion of Life - Disney Animation

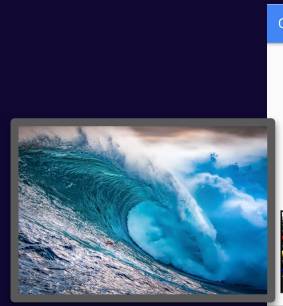
# UI Components

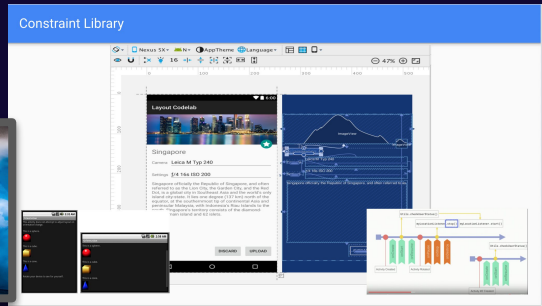
Android Material Components - Create beautiful apps with modular and customizable UI components.

Android Asset Studio & Tools - A collection of tools to easily generate assets such as launcher icons

ConstraintLayout - allows you to create large and complex layouts with a flat view hierarchy

MotionLayout





### App Bars
A flexible toolbar designed to provide a typical Material Design experience.

### Bottom Navigation
Bottom navigation bars make it easy to explore and switch between top-level views in a single tap.

### Bottom Sheets
Bottom sheets slide up from the bottom of the screen to reveal more content.

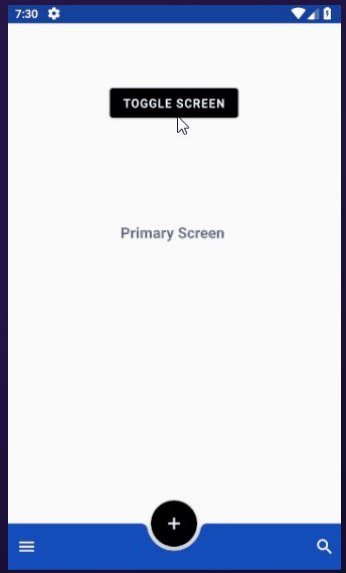
### Collapsing Toolbars
Collapsing toolbars change height and other visual aspects in response to scrolling.

### Coordinated Behaviors
A layout that coordinates interactive behaviors between its children.

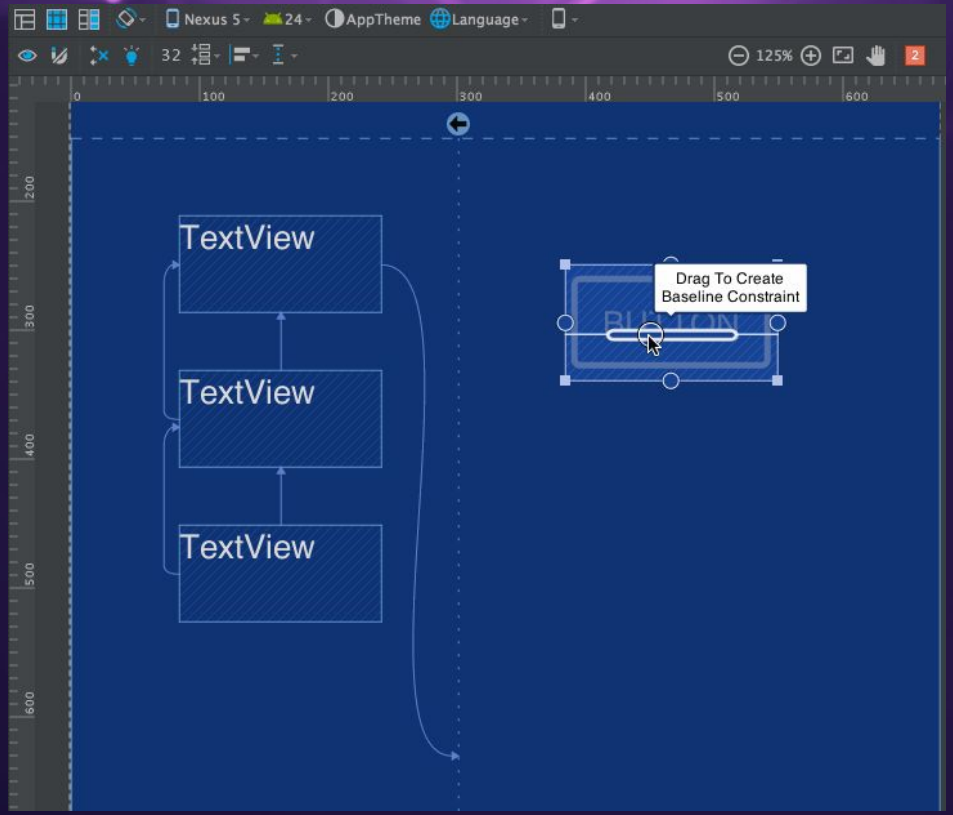
### Floating Action Buttons
A floating button for the primary action in an application.

### Modal Bottom Sheets
Modal bottom sheets act like a dialog at the bottom of the screen.

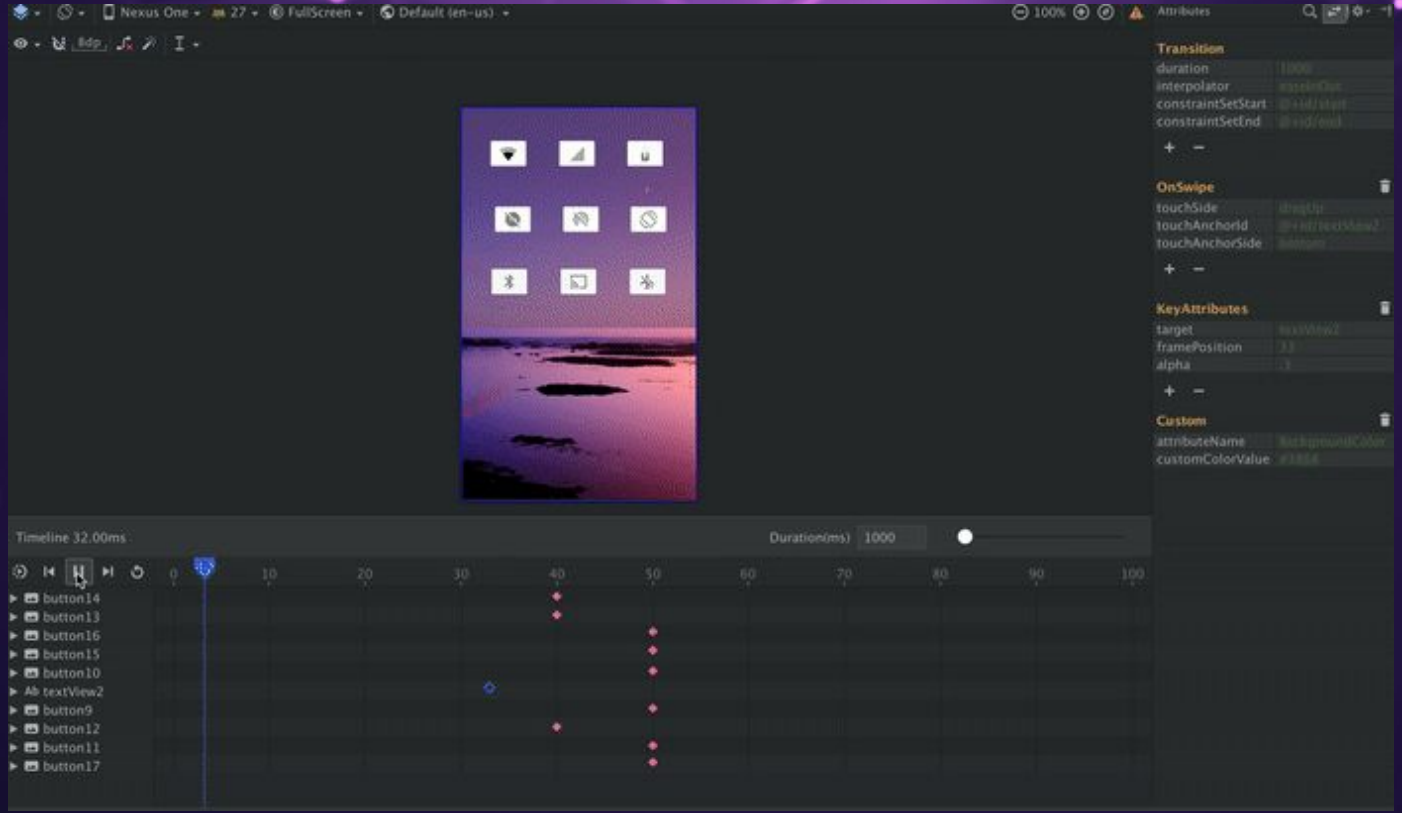
### Navigation Views
A scrollable view that renders a menu resource as a vertical list.



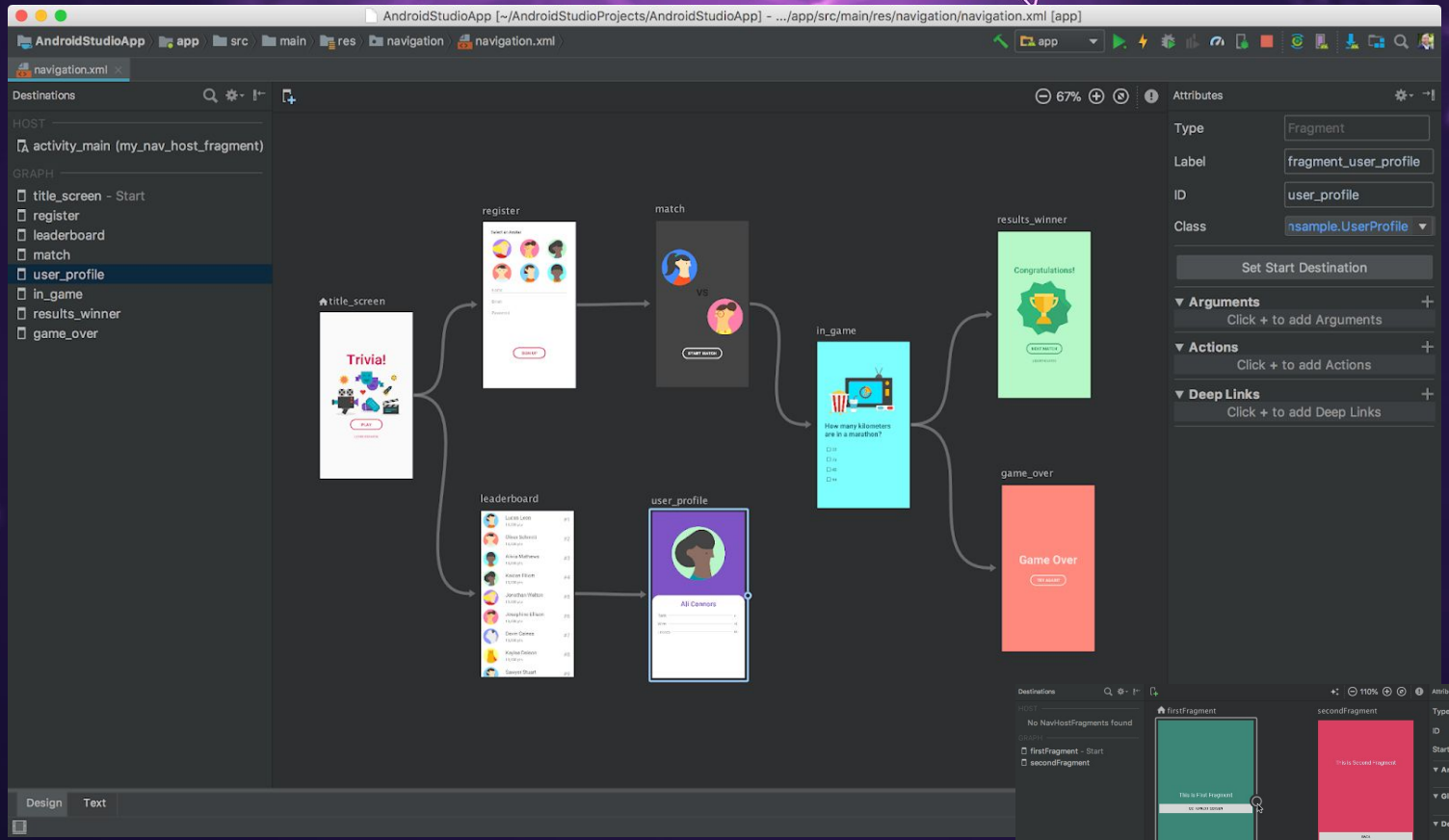

# Material Components




Constraint System



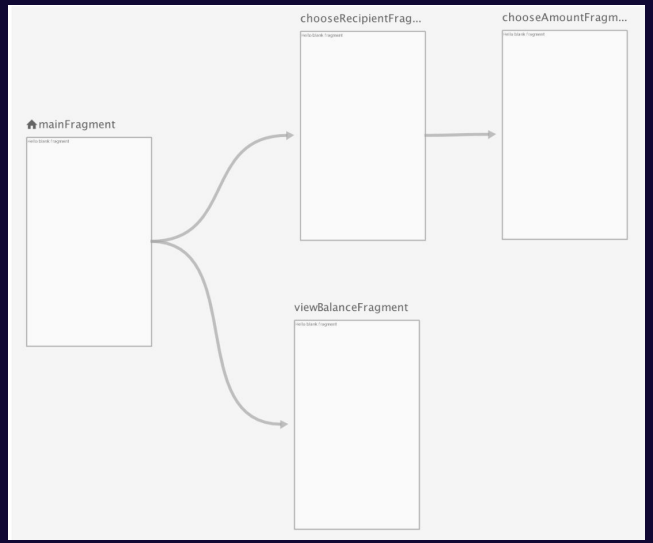

MotionLayout vs After Effects



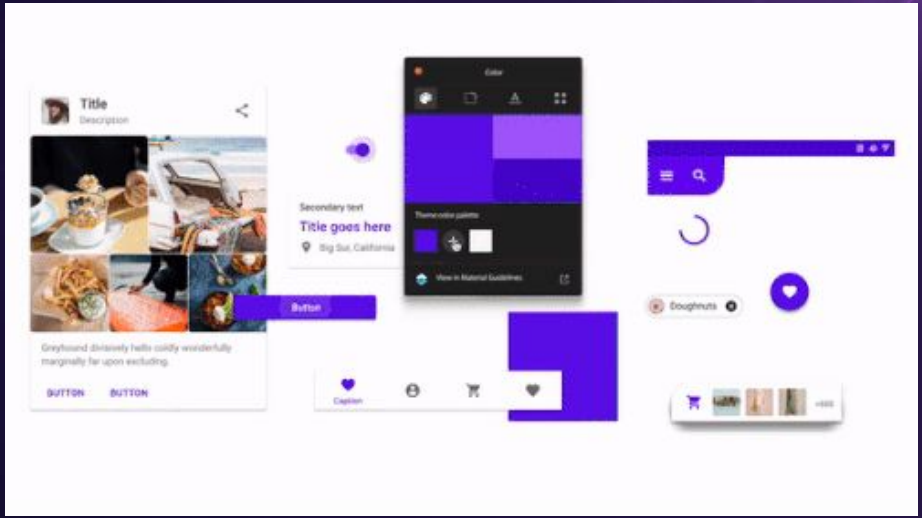

Fast navigation setup and changes

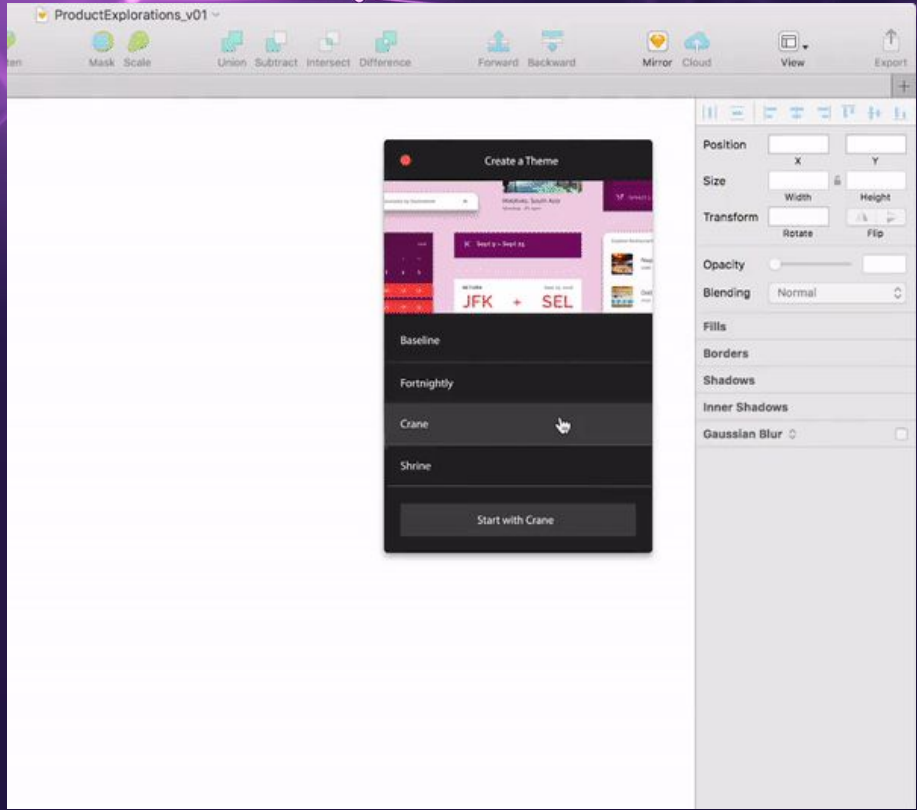
# Navigation in Plain XML

Works great with Git!

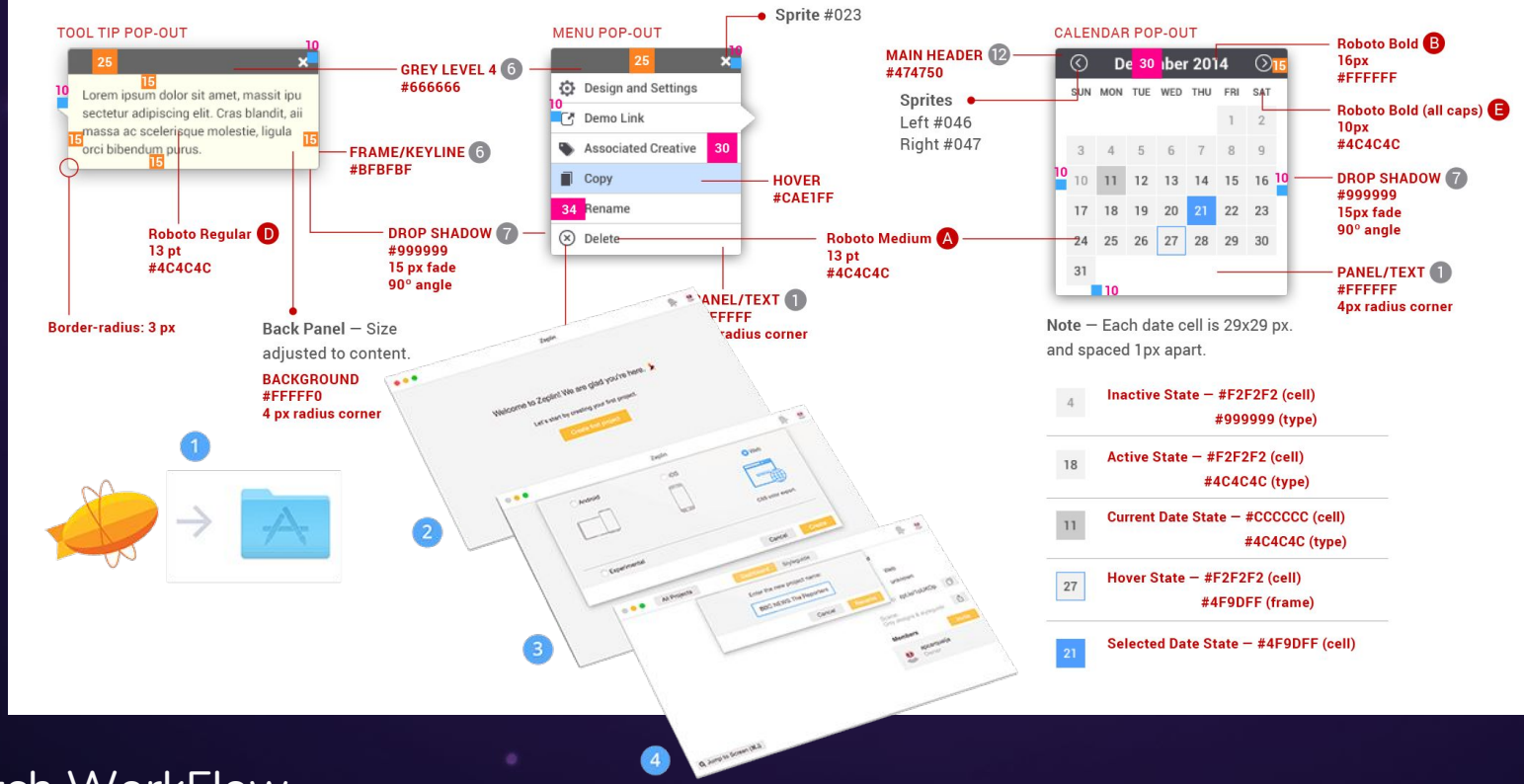



```xml
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    app:startDestination="@id/blankFragment">
    <fragment
        android:id="@+id/blankFragment"
        android:name="com.example.cashdog.cashdog.BlankFragment"
        android:label="fragment_blank"
        tools:layout="@layout/fragment_blank" >
        <action
            android:id="@+id/action_blankFragment_to_blankFragment2"
            app:destination="@id/blankFragment2" />
    </fragment>
    <fragment
        android:id="@+id/blankFragment2"
        android:name="com.example.cashdog.cashdog.BlankFragment2"
        android:label="fragment_blank_fragment2"
        tools:layout="@layout/fragment_blank_fragment2" />
</navigation>
```




# Material Theming




# Sketch WorkFlow

# Library Resources

Awesome resources:

The best Android Resources in each category.

https://snowdream.github.io/awesome-android/

Awesome UI

https://github.com/wasabeef/awesome-android-ui

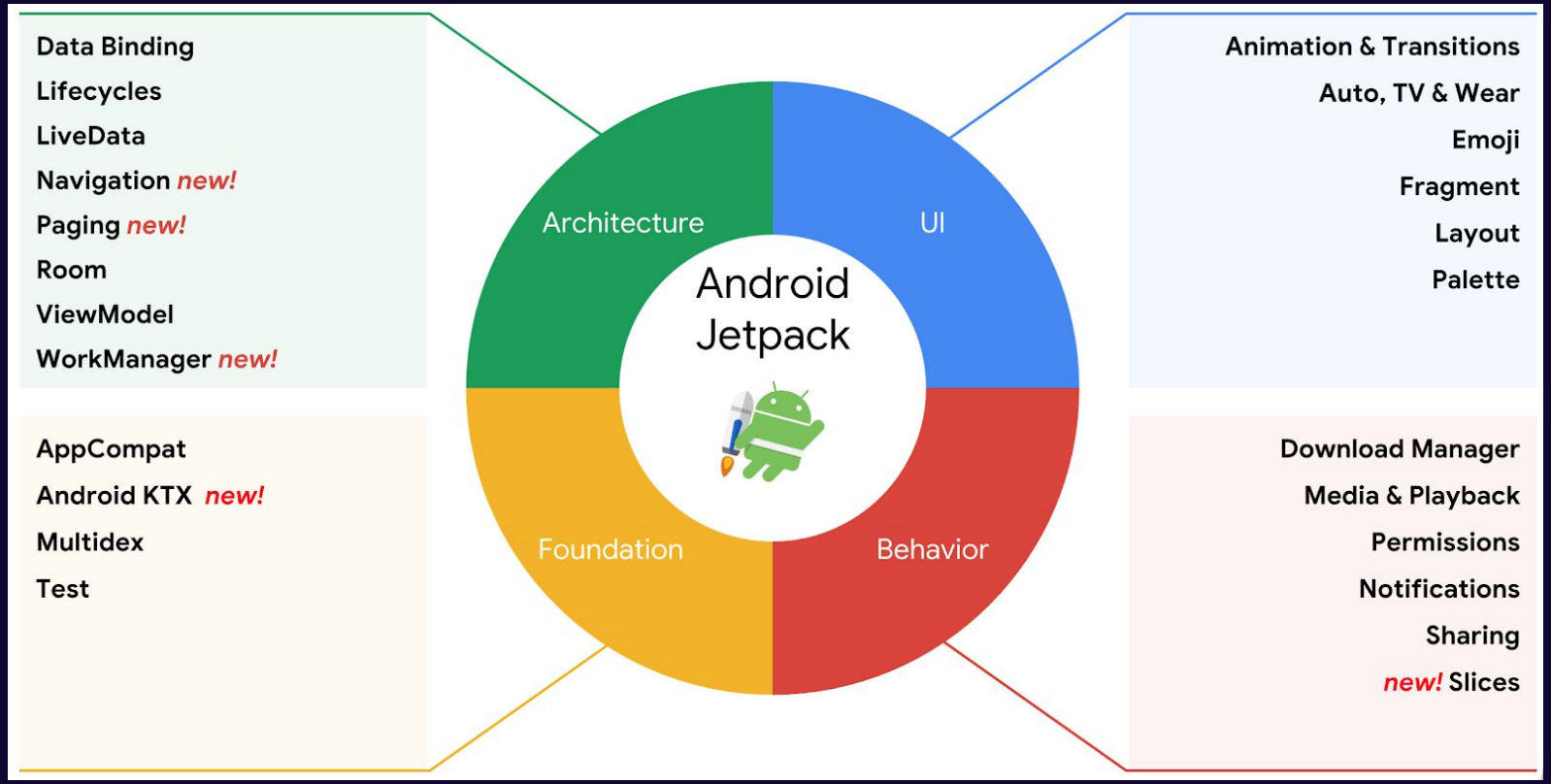
CodePath Must-Have Libs

https://guides.codepath.com/android/Must-Have-Libraries

**Code Design**

To do the **thing right**

**Easily maintain & expand**


Data Binding
Lifecycles
LiveData
Navigation new!
Paging new!
Room
ViewModel
WorkManager new!
AppCompat
Android KTX new!
Multidex
Test
Architecture
UI
Android Jetpack
Foundation
Behavior
Animation & Transitions
Auto, TV & Wear
Emoji
Fragment
Layout
Palette
Download Manager
Media & Playback
Permissions
Notifications
Sharing
new! Slices

# Building Android

# Android Community

Get inspiration from developer stories

https://developer.android.com/distribute/stories/index.html

Keeping Updated with Developer Community

https://guides.codepath.com/android/Keeping-Updated-with-Android

https://medium.com/@magdamiu/lets-talk-android-1f2c588d8726

# Coding Android

Developing for Android - Even though Android uses the Java programming language, the way that code should be written on Android is significantly different than how code in that language might be written for back-end or desktop applications, given the constraints and problems of mobile devices that are very different from those other environments. Most are solved with the introduction of the Android Architecture Components.

Static Analysis - Android Lint , SonarLint & FindBugs

Facebook Infer with Gradle Plugin

GitHub code review tools.

*Use Android Studio to format source!

# What Happened? & Why Important?

- First time the Android Team is proposing an architecture for Android.
- First time the Android Team is supporting a new language.
- Android Team adds UI Tools
- Android Material Design released Material Components

# Kotlin & Android

The State Of the Union

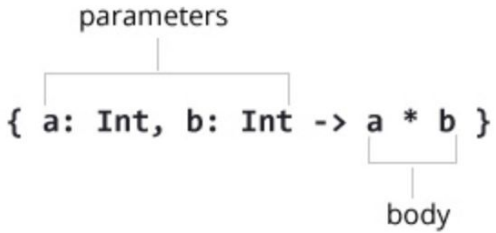
# Swift vs Objective C(rap)

```
- (void)letsMakeASandwich:(void (^)(void))sandwichIngredientsBlock
{
    NSLog(@"Bread");
    sandwichIngredientsBlock();
    NSLog(@"Bread");
}
```

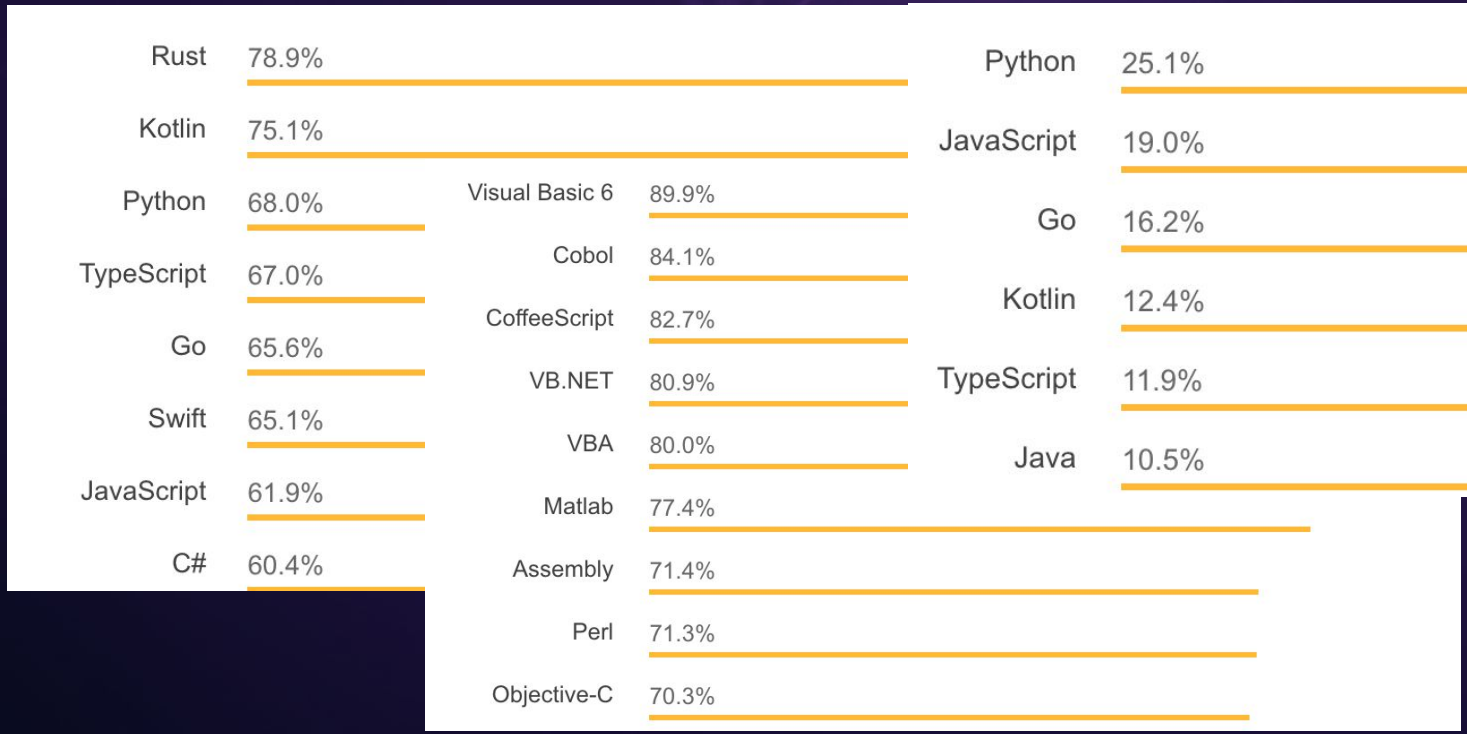
## Closure Expression Syntax

Closure expression syntax has the following general for

```
{ (parameters) -> return type in
    statements
}
```



# Language Matters



# Kotlin

Kotlin fixes a series of issues that Java suffers from

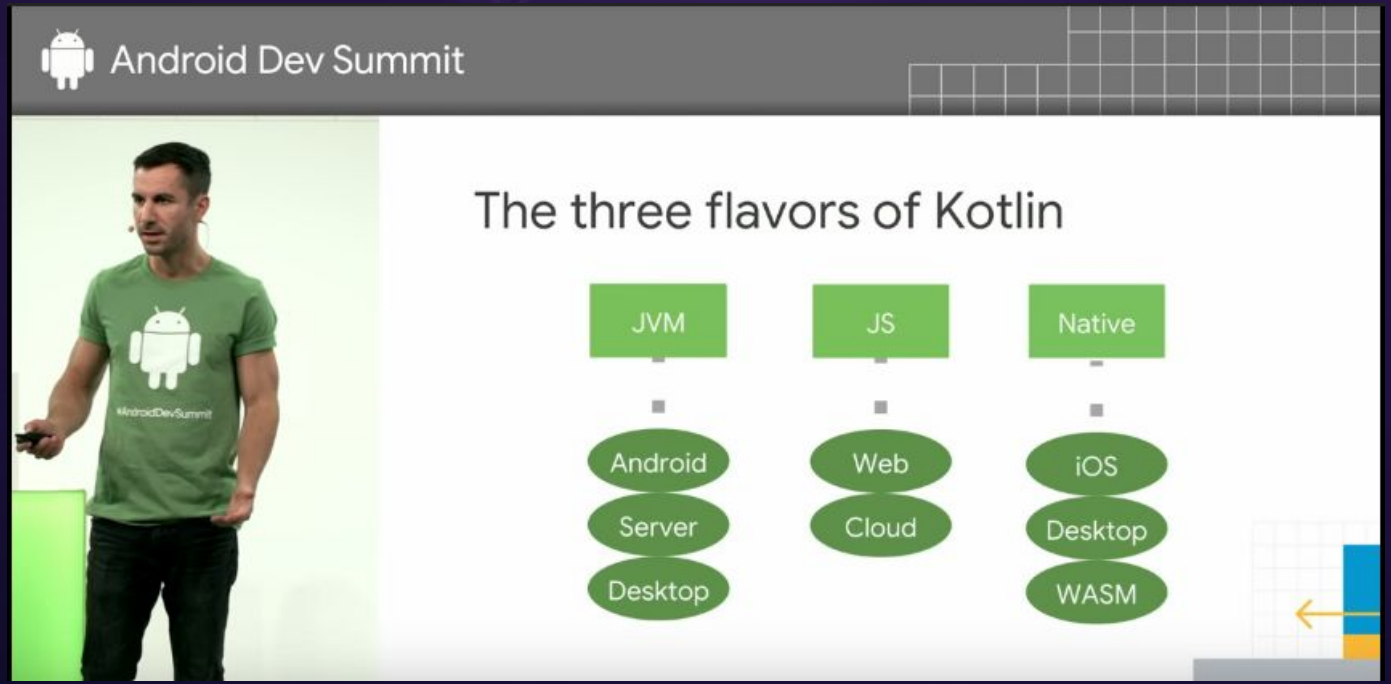
# Kotlin/Swift Features

- Open Source
- Lambda expressions - Higher Order Functional Language
- Kotlin is null safe by default
- Extension functions
- Coroutines are first-class citizens
  - Concurrency Is Not Parallelism*
- There are no checked exceptions
- Native support for delegation
- Default Parameters
- Data classes
- Smart casts
- Support for primary & secondary constructors
- Interchangeability with Java
- IDE Support
- Solid Design and Fast Evolution

*“**Concurrency** is about dealing with lots of things at once. **Parallelism** is about doing lots of things at once.” — **Rob Pike**.

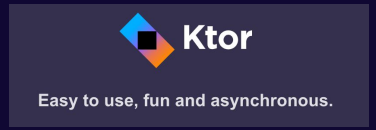
# One Language to Rule Them All

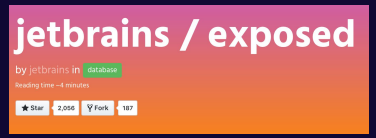
Wojtek

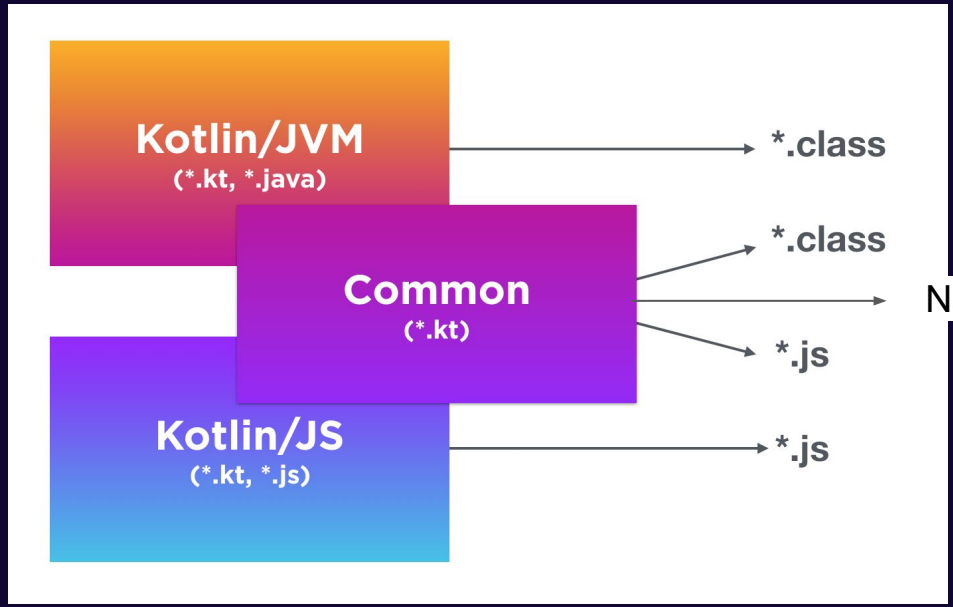


# Kotlin Multi-Platform

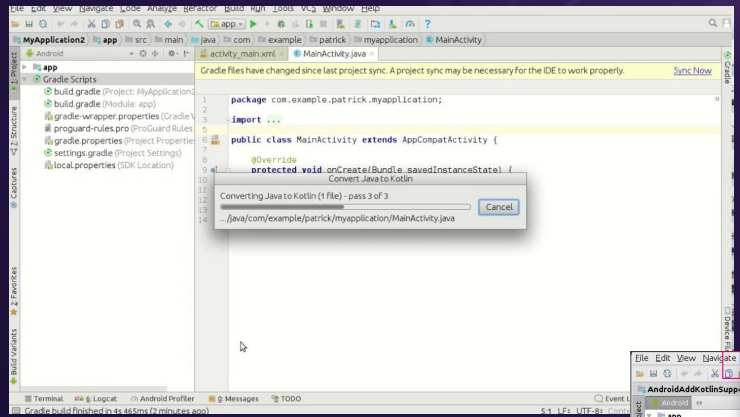
- Kotlin JVM
- Kotlin JS
- Kotlin Native

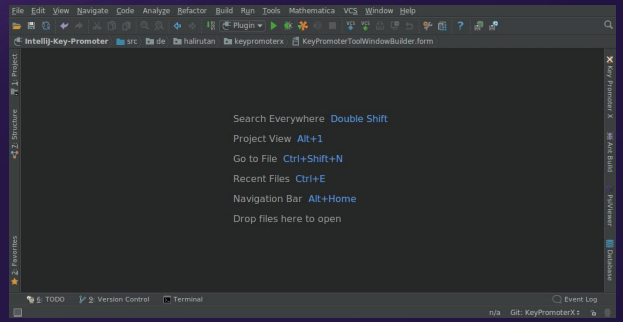


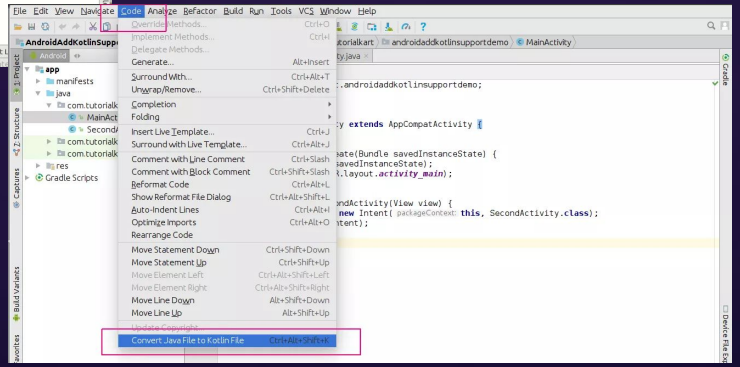



# Android Studio with Kotlin



# Kotlin KTX

https://www.youtube.com/watch?v=LP3PaPrIFHo

1. KTX does not add anything.  Only clean up the Kotlin calls.
2. Use Kotlin spec lang feature
3. Long term useful
4. Meaning upgrade
5. Intent of the extension clear

https://github.com/Kotlin/KEEP/issues/110

# Android KTX

| Module (artifact) | Version | Package |
|---|---|---|
| androidx.core:core-ktx | 1.0.0 | See all the core packages below. |
| androidx.fragment:fragment-ktx | 1.0.0 | androidx.fragment.app |
| androidx.palette:palette-ktx | 1.0.0 | androidx.palette.graphics |
| androidx.sqlite:sqlite-ktx | 2.0.0 | androidx.sqlite.db |
| androidx.collection:collection-ktx | 1.0.0 | androidx.collection |
| androidx.lifecycle:lifecycle-viewmodel-ktx | 2.0.0 | androidx.lifecycle |
| androidx.lifecycle:lifecycle-reactivestreams-ktx | 2.0.0 | androidx.lifecycle |
| android.arch.navigation:navigation-common-ktx | 1.0.0-alpha06 | androidx.navigation |
| android.arch.navigation:navigation-fragment-ktx | 1.0.0-alpha06 | androidx.navigation.fragment |
| android.arch.navigation:navigation-runtime-ktx | 1.0.0-alpha06 | androidx.navigation |
| android.arch.navigation:navigation-testing-ktx | 1.0.0-alpha06 | androidx.navigation.testing |
| android.arch.navigation:navigation-ui-ktx | 1.0.0-alpha06 | androidx.navigation.ui |
| android.arch.work:work-runtime-ktx | 1.0.0-alpha10 | androidx.work.ktx |

androidx.core:core-ktx

KOTLIN | KOTLIN + ANDROID KTX

```
sharedPreferences.edit {
    putBoolean("key", value)
}
```

KOTLIN | KOTLIN + ANDROID KTX

```
view.doOnPreDraw {
     actionToBeTriggered()
}
```

androidx.sqlite:sqlite-ktx

KOTLIN | KOTLIN + ANDROID KTX

```
db.transaction {
    // insert data
}
```

androidx.fragment:fragment-ktx

KOTLIN | KOTLIN + ANDROID KTX

```
supportFragmentManager.transaction(allowStateLoss = true) {
            replace(R.id.my_fragment_container, myFragment, FRAGMENT_TAG)
        }
```

# KTX Android "Discoverability"

**Romain Guy**
@romainguy

Follow

One of my favorite android-ktx extension is createBitmap() because it defaults the config to ARGB_8888 github.com/android/androi

# 31 Days of Kotlin

https://twitter.com/i/moments/980488782406303744

# Android Lint (Kotlin)


1. False positives over false negatives
    a. Ability to suppress issues
    b. Long running time over missing errors
2. Focus on Android (no longer true!)

Features

- Check Java / Kotlin / XML / AndoirdManifest / gradel / .class / .pro / .cfg / .png  etc ...
- Static checker & 75% Runs in the editor with Quick Fixes!
- Deep explanation into what Lint is describing as the error
- Runs with on release and CI/CD
- Custom checks (Lint API is not stable) in Kotlin

# Kotlin DI / SL


```
class MyViewModelActivity : AppCompatActivity() {

    // Lazy Inject ViewModel
    val myViewModel: MyViewModel by viewModel()

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_simple)
```

# Learning Kotlin *{Swift}*

Kotlin language

- Kotlinlang.org: The official Kotlin website. Includes everything from a guide to basic syntax to the Kotlin standard library reference.
- Kotlin Koans Online: A collection of exercises in an online IDE to help you learn the Kotlin syntax.

Kotlin on Android

- Get Started with Kotlin on Android: A short guide to start using Kotlin in Android Studio.

https://developer.android.com/kotlin/resources

# Android Architecture

## Android Architectural Components

# We have good base ... not a framework!

"Android has good bones"

- Stable, compact set of fundamental concepts
- Strong contracts between apps and OS
- Enables OS to actively manage applications
- Supports a broad range of app models & frameworks
- Supports a broad range of devices with a unified programming model

# App components

App components are the essential building blocks of an Android app ??? NO !!!

- **Activities** - The entry point for interacting with the user.
- **Services** - General-purpose entry point for keeping an app running in the background.
- **Broadcast receivers** - Component that enables the system to deliver events to the app outside of a regular user flow
- **Content providers** - Manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access.

# The Problem

"How should I design my Android application?  What kind of MVC pattern should I use?  What should I use for an event bus?" “

It is probably better to call the core Android APIs a ‘system framework.’  For the most part, the platform APIs we provide are there to define how an application interacts with the operating system; but for anything going on purely within the app, these APIs are often just not relevant.” -  Dianne Hackborn

https://plus.google.com/+DianneHackborn/posts/FXCCYxepsDU

# Problem Resolution → Solution

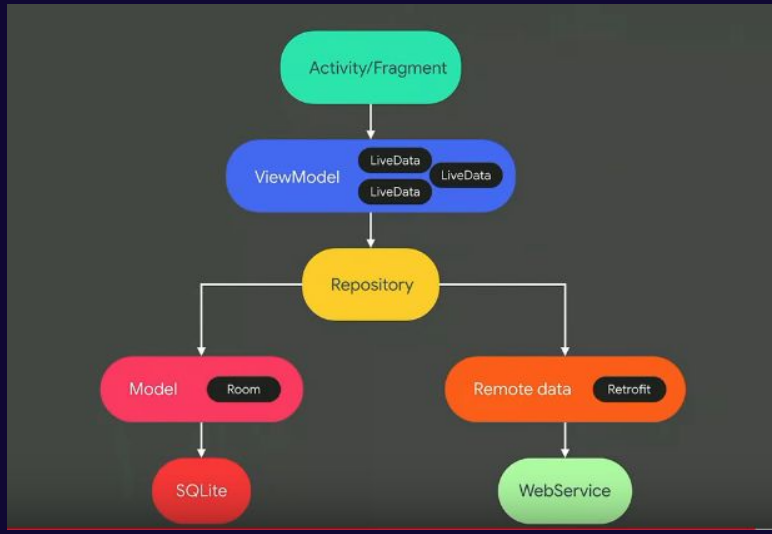
Android Architecture Components work together to implement a sane app architecture, while they individually address developer pain points.

Pieces needed to finish the platform:

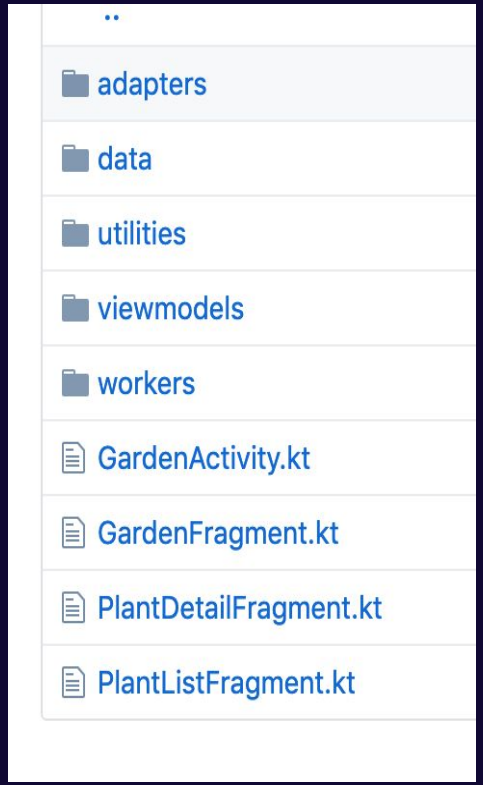
- Persist Data
- Manage Lifecycles
- Modular
- Defence Against Common errors
- Less Boiler Plate

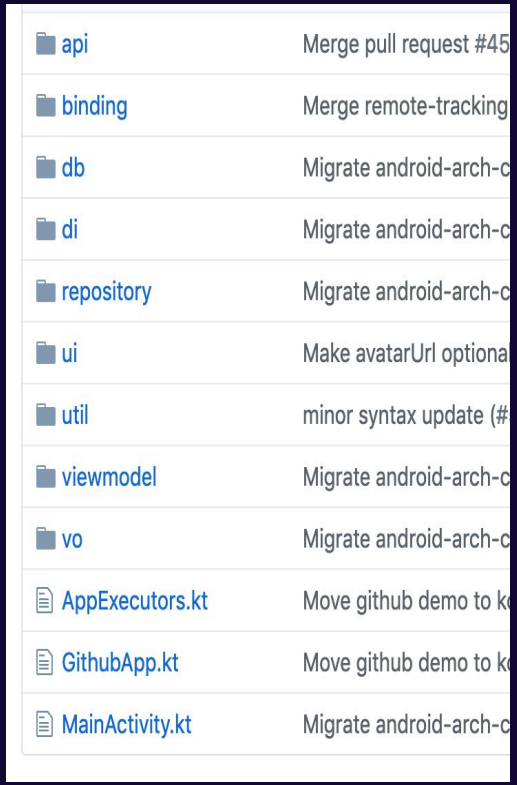
# A Modern Android App.

- Kotlin (Coroutines not RxJava)
- Service Locator (Koin) / Dependency Injector
- Android KTX (AndroidX namespace)
- Android Architectural Components
  - Lifecycle
  - ViewModel
  - LiveData with two way Data Binding
  - Room / Paging Lib
- WorkManager
- Navigation & MotionLayout Editor
- Material Components
- Firebase backend / Retrofit (REST)
- Moshi not GSON



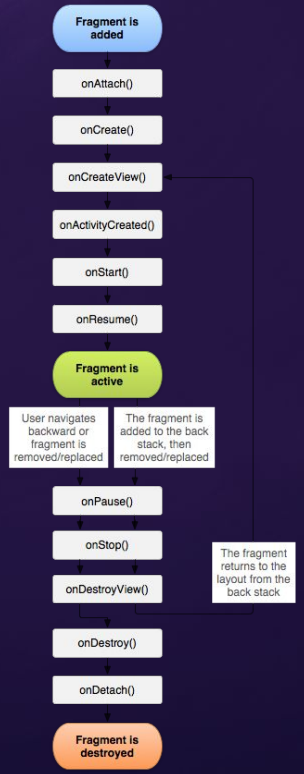
# Android App Directory Structure.

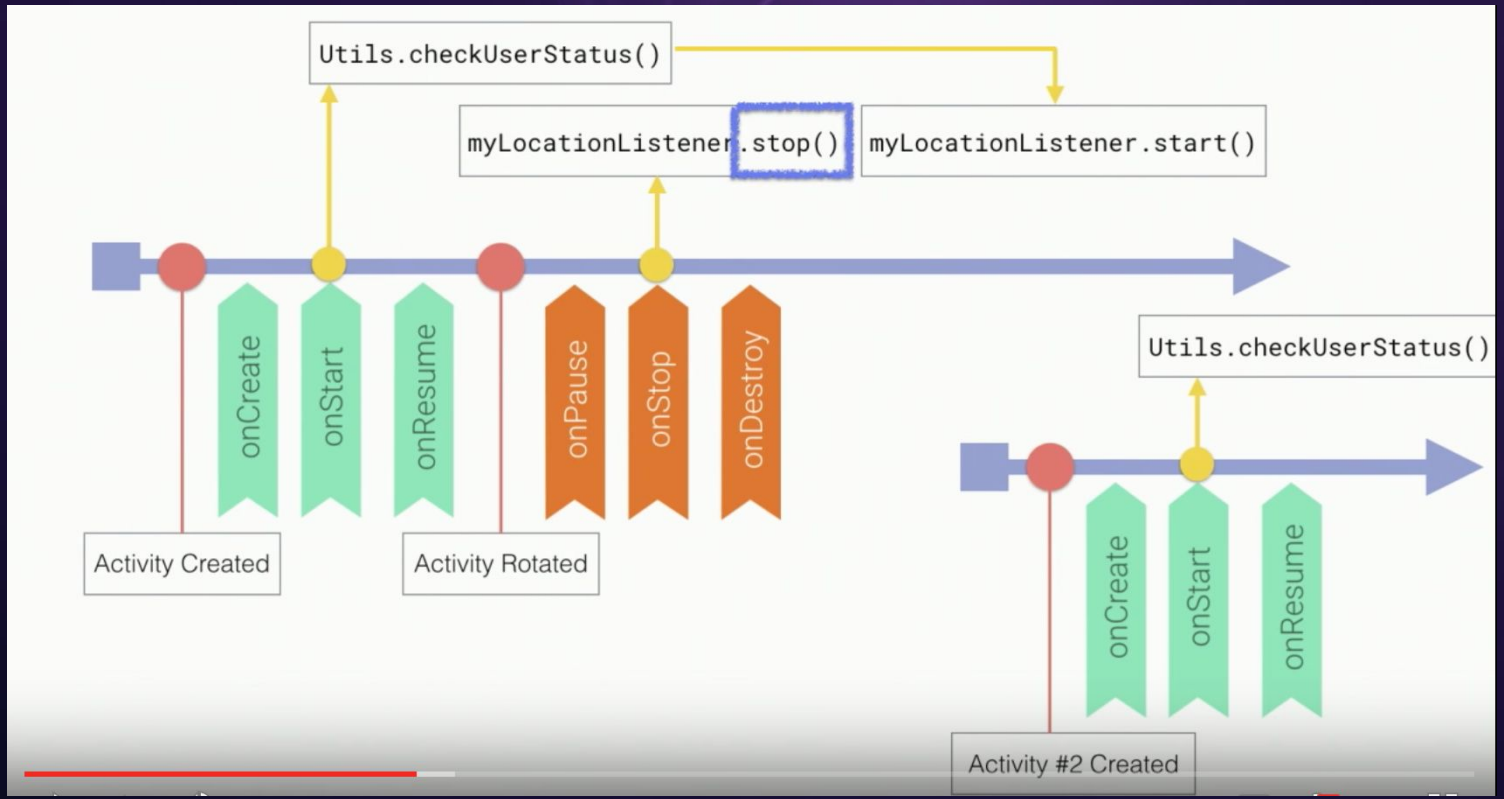



# Lifecycle Aware Components

# Android Activity LifeCycle / Fragment LifeCycle

# Rotate Phone

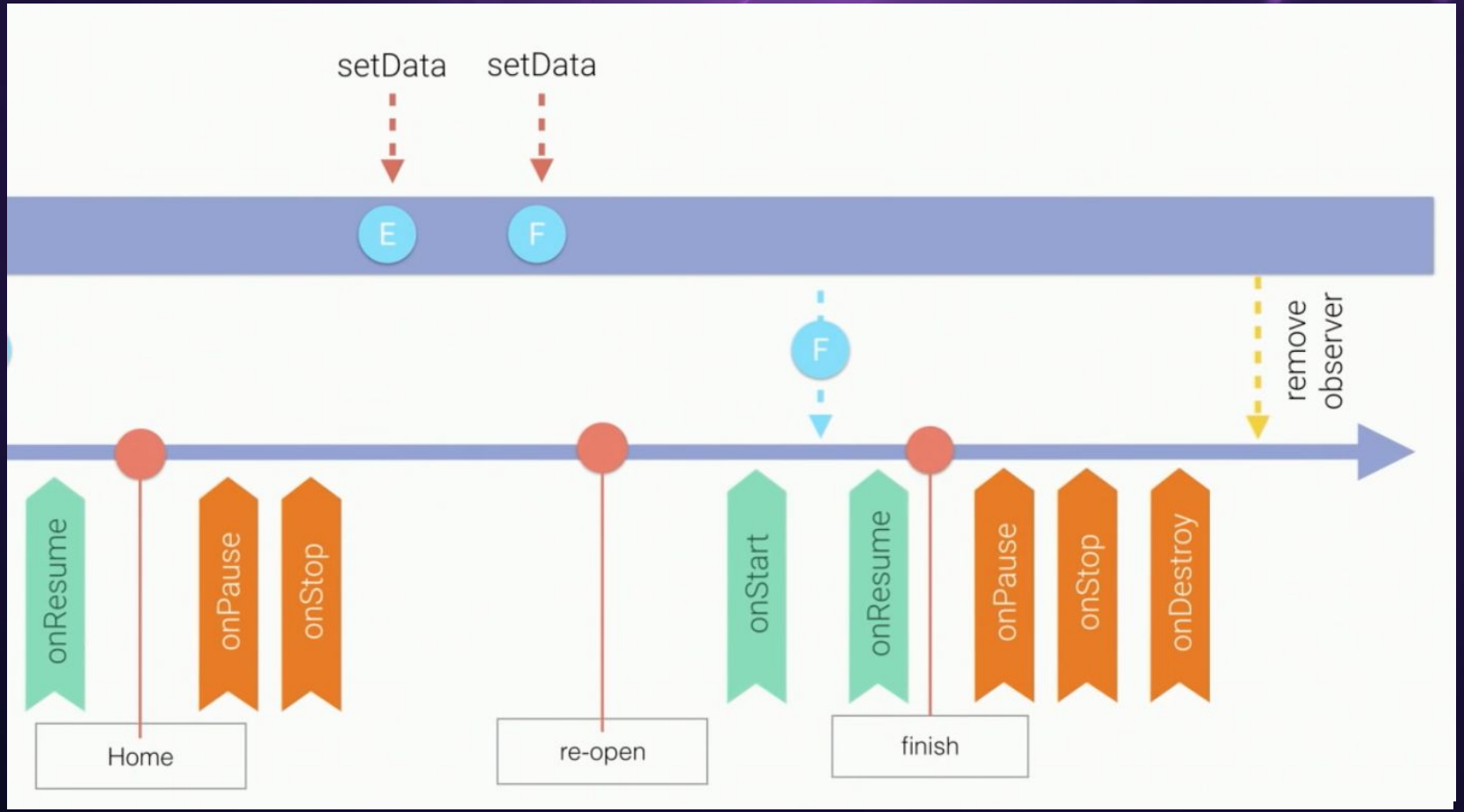


# LifeCycles (Activity & Fragments)

Lifecycle is a class that holds the information about the lifecycle state

- Keep your UI controllers (activities and fragments) as lean as possible.
- Try to write data-driven UIs with ViewModel & LiveData.
- Put your data logic in your ViewModel class.
- Use Data Binding to maintain a clean interface between your views and the UI controller.
- Create a Presenter class to handle UI modifications.
- Never reference a View or Activity context in your ViewModel.

# LiveData Flow

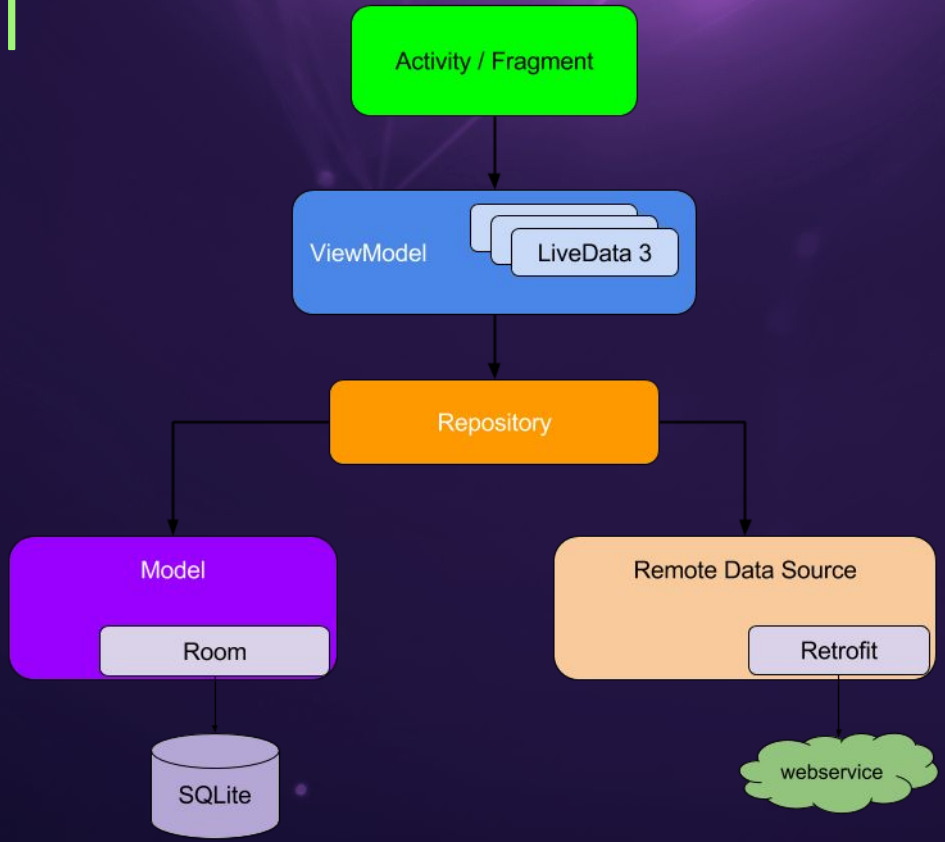


# Advantages of LiveData

LiveData is a data holder class that keeps a value  and is observed.

- No memory leaks
- No crashes due to stopped activities
- Always up to date data
- Proper configuration change
- Sharing Resources
- No more manual lifecycle handling

# ViewModel

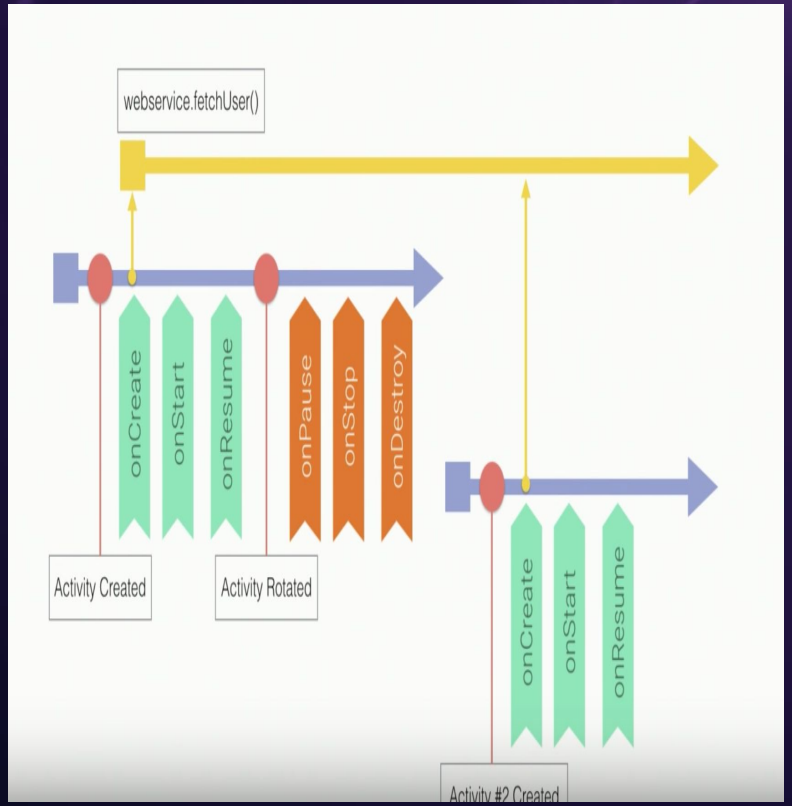

Activity / Fragment
ViewModel
LiveData 3
Repository
Model
Room
SQLite
Remote Data Source
Retrofit
webservice

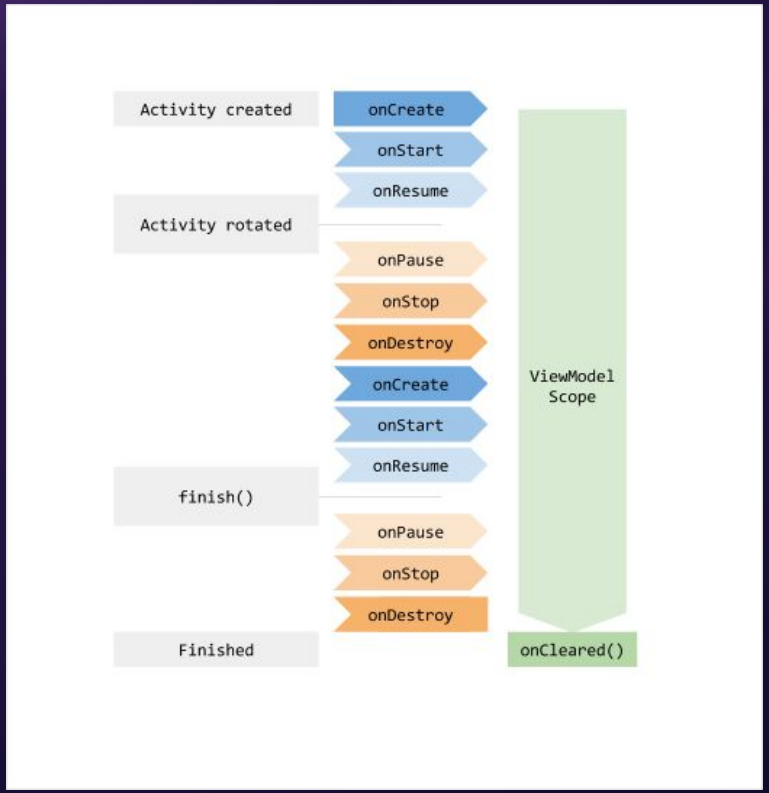
# ViewModel

Provide data for UI components and survive config change

- Lifecycles provides a new class called ViewModel, a helper class for the UI controller which is responsible for preparing the data for the UI.
- ViewModels usually expose this information via LiveData or Android Data Binding.
- LiveData handles the notification side of things while the ViewModel makes sure that the data is retained appropriately.
- The *lifecycle-viewmodel-savedstate* access saved state

# ViewModel


webservice.fetchUser()
onCreate
onStart
onResume
onPause
onStop
onDestroy
Activity Created
Activity Rotated
onCreate
onStart
onResume
Activity #2 Created


Activity created
onCreate
onStart
onResume
Activity rotated
onPause
onStop
onDestroy
onCreate
onStart
onResume
finish()
onPause
onStop
onDestroy
Finished
ViewModel Scope
onCleared()

# LiveData Binding ViewModel

```
findViewById<TextView>(R.id.plain_name).apply {
    text = viewModel.userName
~~~
<layout ...>
    <data>
        <variable
            name="viewmodel"
            type="com.myapp.data.ViewModel" />
    </data>
    <ConstraintLayout... /> <!-- UI layout's root element -->
    <TextView
        android:id="@+id/plain_name"
        ...
        android:text="@={viewmodel.name}"
        android:onClick="@{() -> viewmodel.onLike()}"/>
</layout>
```

# Sharing Data Between Fragments

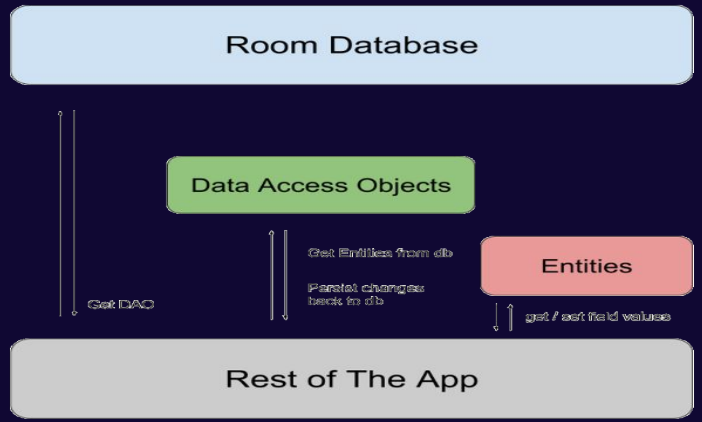
- The activity does not need to do anything, or know anything about this communication.
- Fragments don’t need to know about each other besides the SharedViewModel contract. If one of them disappears, the other one keeps working as usual.
- Each fragment has its own lifecycle, and is not affected by the lifecycle of the other one. In fact, in a UI where one fragment replaces the other one, the UI works without any problems.

# Room Persistence Library

Room provides an abstraction layer over SQLite to allow fluent database access while - harnessing the full power of SQLite.

3 major components in Room:

- Database:
- Entity:
- DAO:



# ROOM Layout

```java
@Dao
public interface MyDao {
    @Query("SELECT first_name, last_name FROM user WHERE region IN (:regions)")
    public LiveData<List<User>> loadUsersFromRegionsSync(List<String> regions);
}
```

```java
@Entity
class Feed {
  @PrimaryKey
  int id;
  String title;
  String subTitle;
}
```

```java
@Database(entities = {Feed.class}, version = 1)
abstract class MyDatabase extends RoomDatabase {
  abstract FeedDao feedDao();
}
```

# Android App Architecture

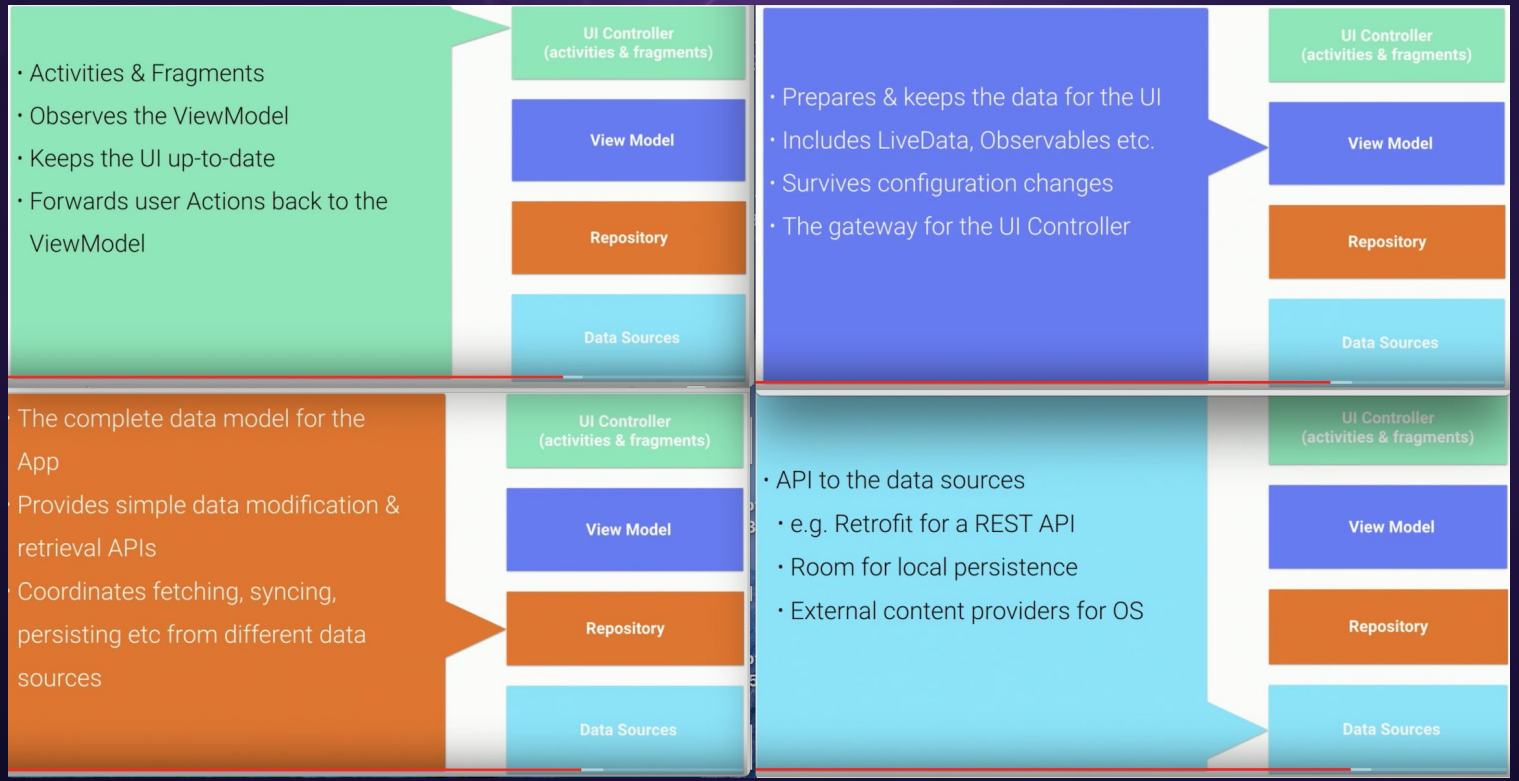
# Arch Guiding principles

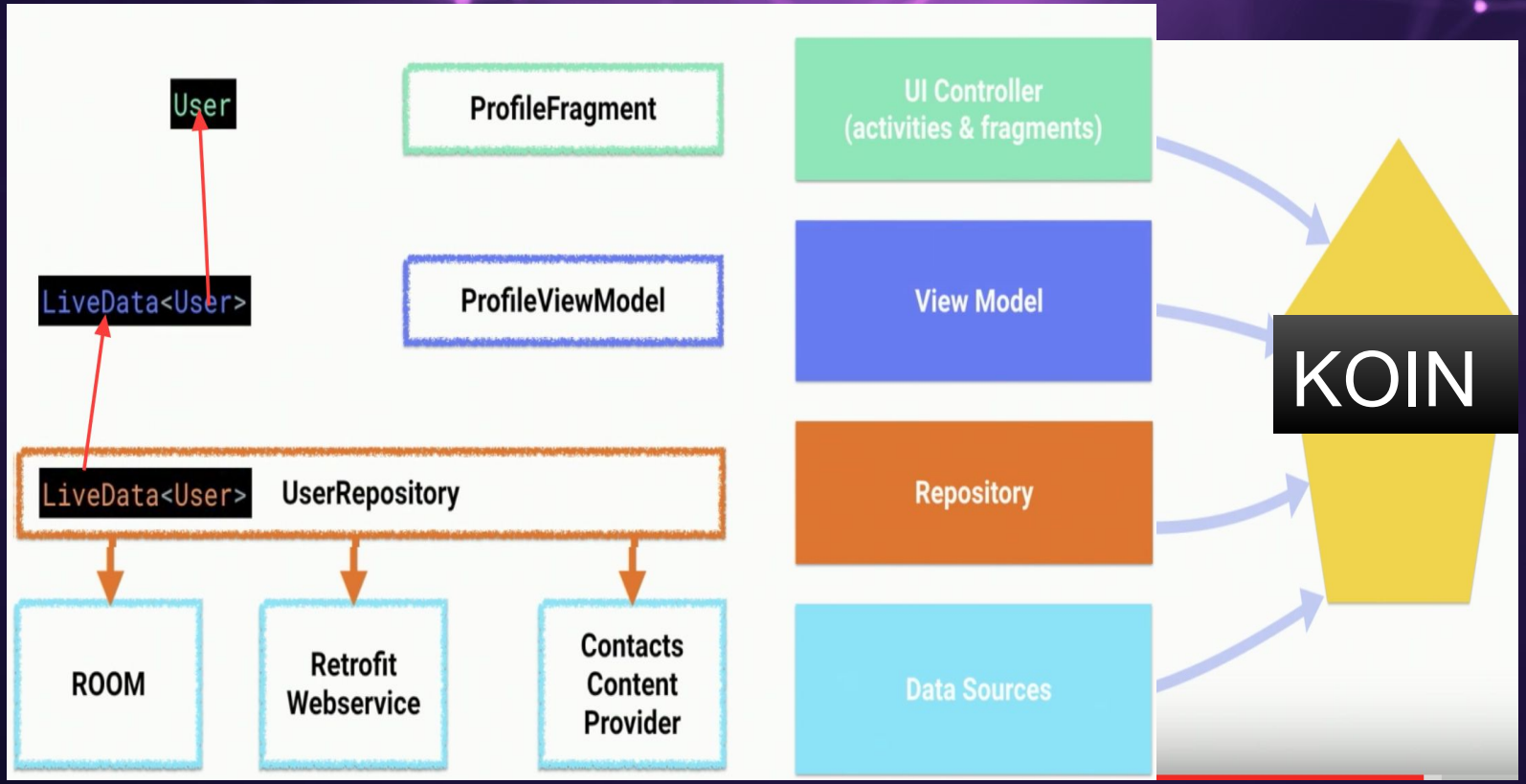
- The entry points you define in your manifest - activities, services, broadcast receivers, etc. - are not the source of data.
- Be merciless in creating well defined boundaries of responsibility between various modules of your app.
- Expose as little as possible from each module
- As you define the interaction between the modules, think about how to make each one testable in isolation.

# Arch Guiding principles

- The core of your app is what makes it stand out from the rest.
- Persist as much relevant and fresh data as possible so that your app is usable when the device is in offline mode.
- Your repository should designate one data source as the single source of truth.  -- For more information, see Single source of truth.

# Architectural components






Testable Reactive Architecture

# CodeLabs & GitHub.

This codelab introduces you to architecture components:

https://codelabs.developers.google.com/

Github:

https://github.com/googlesamples/android-architecture-components

# TDD

User Interface & Interactions:

- Android UI Instrumentation test.Espresso test.

ViewModel & User Repository

- The ViewModel can be tested using a JUnit test.

ROOM & Dao:

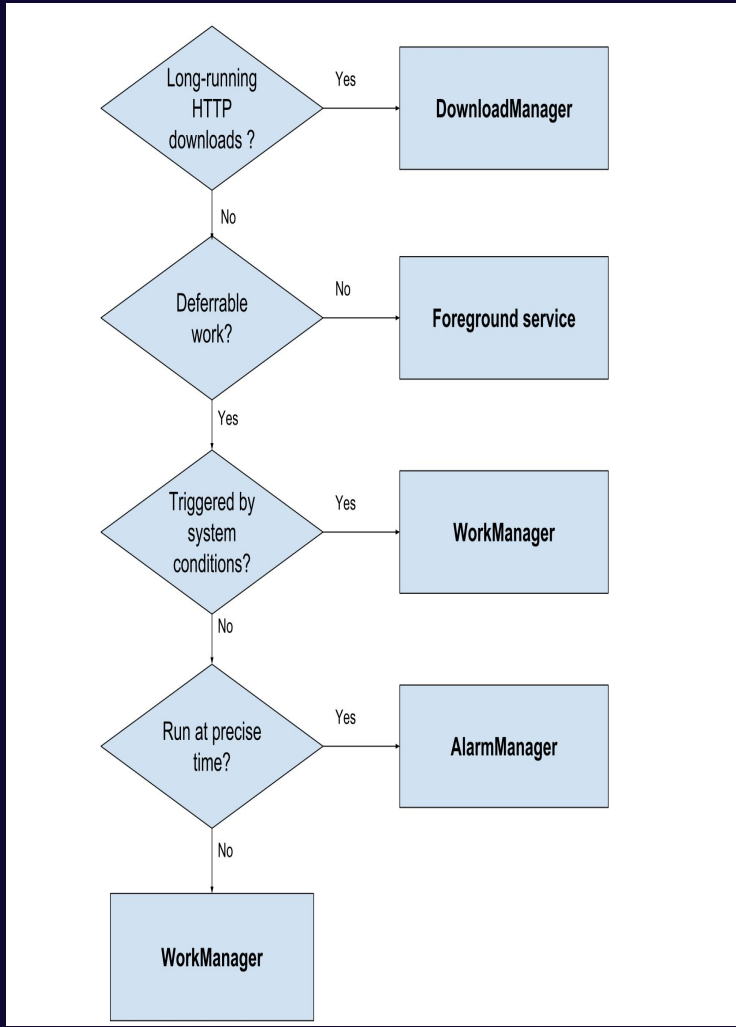
- Using instrumentation tests with in-memory database

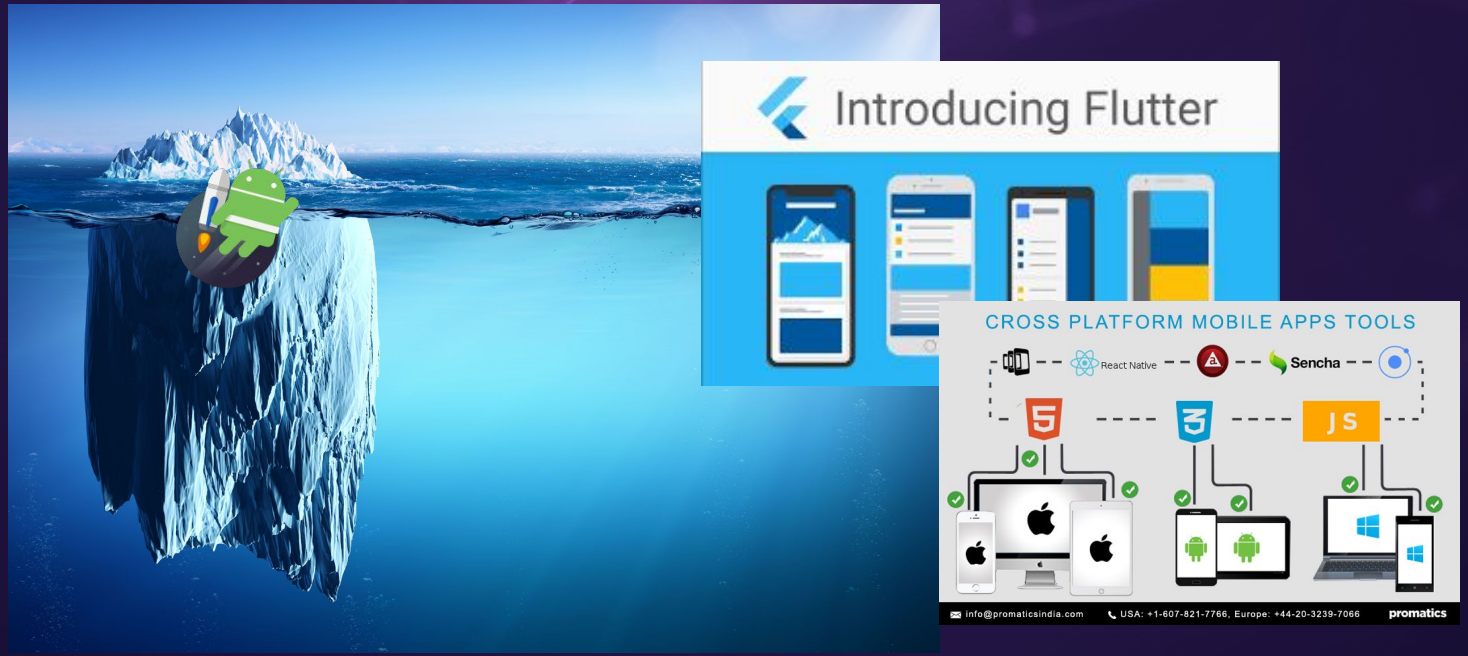
# WorkManager

Schedule tasks

The WorkManager API makes it easy to specify deferrable, asynchronous tasks and when they should run.  It might use JobScheduler, Firebase JobDispatcher, or AlarmManager

You let it choose the best option.






We are just getting started ...

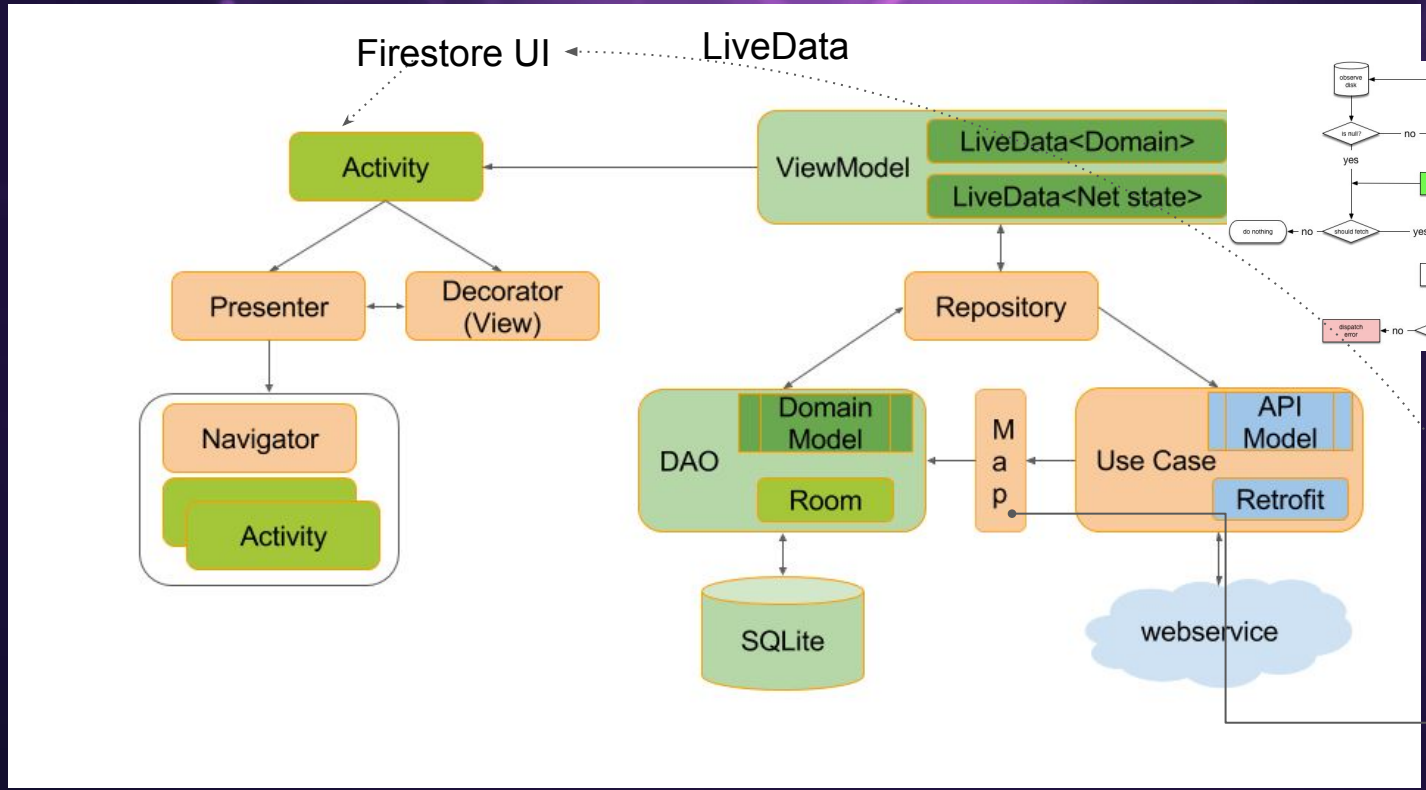
https://developer.android.com/topic/libraries/architecture/viewmodel-savedstate

# Firebase components backend

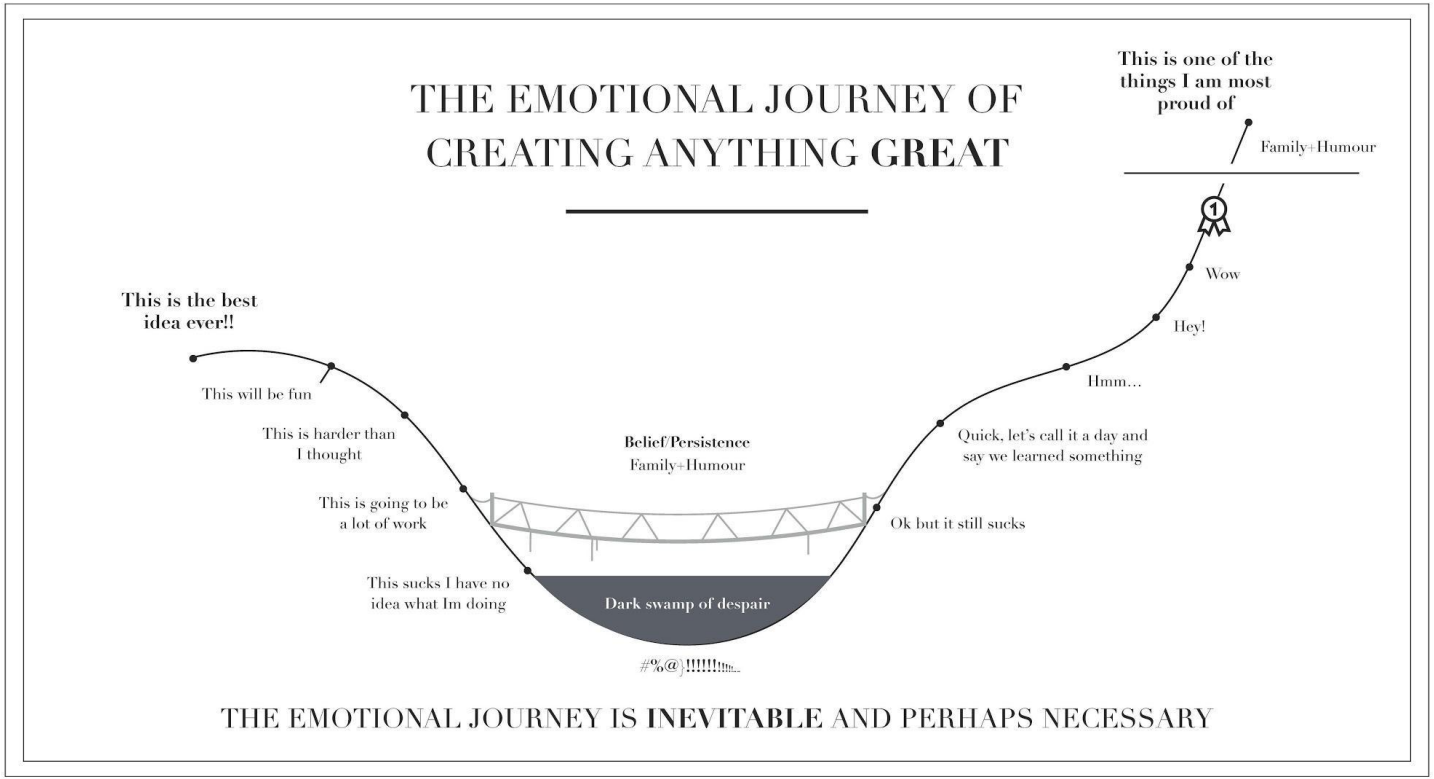
Firebase is amazing at building a synchronized backend that scales like king kong in record time!

Using Firebase realtime DB with the Android Architecture Components (3 parts)

Review Source Code!




Final networked component architecture




@biocodes

www.zoewave.com