What's new on Android Jetpack

@Compose with Declarative UI

Muh Isfhani Ghiath

find me on @isfaaghyth

Currently
Software Engineer - Android, **Tokopedia**
Country Representative, **Google Crowdsource**
Co-Organizer, **GDG Jakarta**

Previously
Community Lead, **Developer Student Clubs**
Founder, **daeng.id**
CTO, **Frypto Ltd**
Campus Expert, **Github**
Head of Engineering, **planetwin.co.id**
CTO, **PT Startup Digital Indonesia**

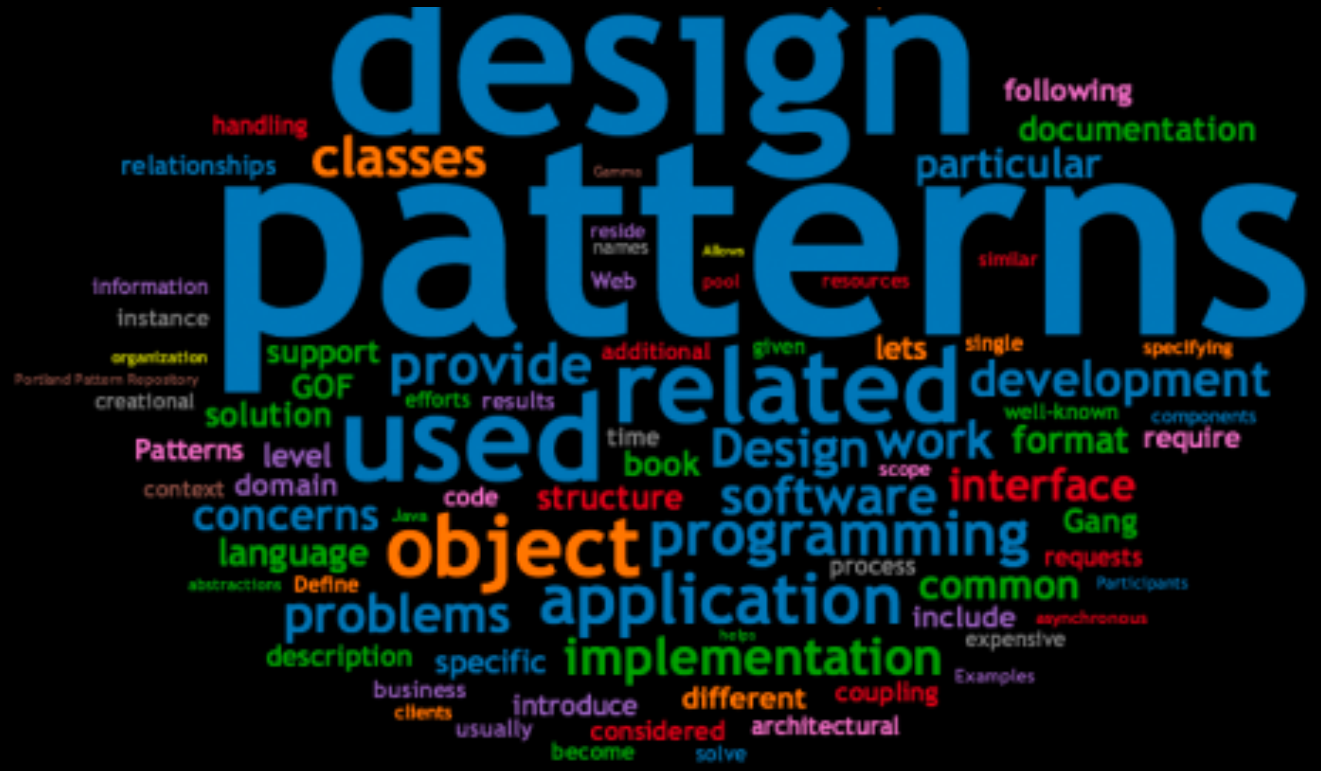

Shoreline
AMPHITHEATRE at MOUNTAIN VIEW

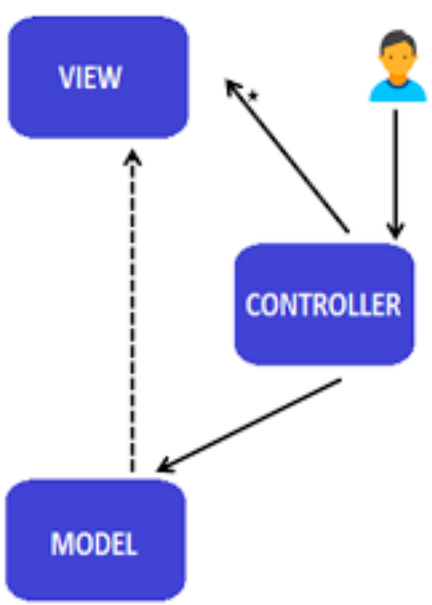

Google
1945 & 1965 Charleston Road


Google

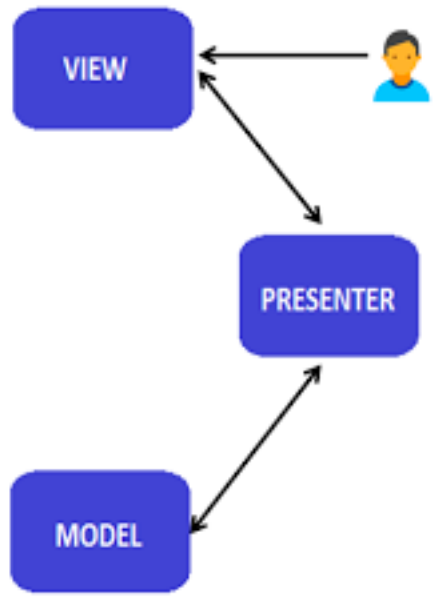

Shoreline
AMPHITHEATRE at MOUNTAIN VIEW

Jetpack? Declarative UI? Compose?

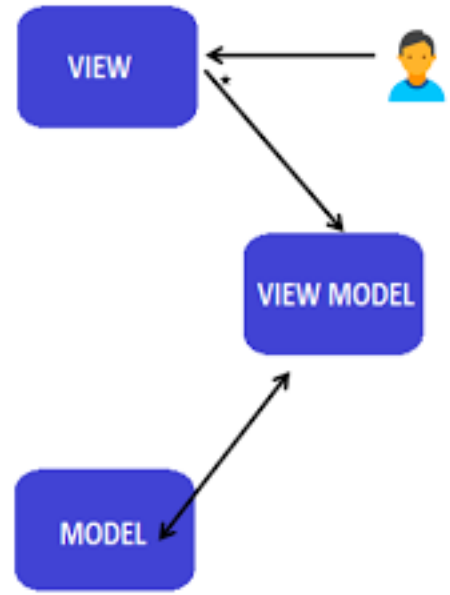

design
patterns
following
documentation
handling
relationships
classes
particular
Gamma
reside
names
Allows
similar
Web
pool
resources
information
instance
organization
Portland Pattern Repository
support
provide
additional
given
lets
single
specifying
GOF
related
development
creational
efforts
results
well-known
components
solution
used
time
Design
work
format
require
Patterns
level
book
scope
context
domain
code
structure
software
interface
concerns
Java
Gang
language
object
programming
requests
process
abstractions
Define
common
Participants
problems
application
include
asynchronous
helps
expensive
description
specific
implementation
Examples
business
different
coupling
clients
introduce
usually
considered
architectural
become
solve


VIEW
CONTROLLER
MODEL

MVC


VIEW
PRESENTER
MODEL

MVP


VIEW
VIEW MODEL
MODEL

MVVM

Android Jetpack

Jetpack is a suite of libraries, tools, and guidance to help developers write high-quality apps easier. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about.

Jetpack

Accelerate Development

Eliminate Boilerplate code

Build high Quality, robust apps

Jetpack Components

Foundation

Backwards compatibility,
Testing, etc.

Architecture

ViewModel, Room, LiveData
Data Binding, etc.

Behaviour

Notification, Permission,
Sharing, etc.

UI

Animation, Fragment,
Palette, Emoji, **Compose**, etc.

Declarative UI

```
// Imperative style
b.setColor(red)
b.clearChildren()
ViewC c3 = new ViewC(...)
b.add(c3)
```

```
// Declarative style
return ViewB(
  color: red,
  child: ViewC(...),
)
```

Jetpack Compose?

API Regret

View.java

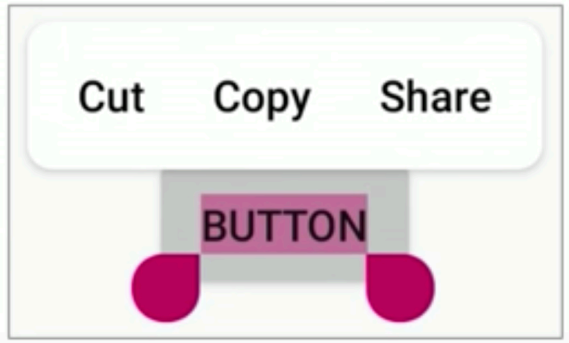
```
27923                     mListenerInfo.mOnHandledKeyLis
27924                     if (mParent instanceof ViewGrou
27925                         ((ViewGroup) mParent).decr
27926                     }
27927                 }
27928             }
27929         }
27930     }
27931 }
```

API Regret

```
public class Spinner extends AbsSpinner {
```

API Regret

```
public class Button extends TextView {
```



Factoring UIs on Android

Fragment ?

Custom View/ViewGroup?

```
public class IsfaGantengWidget extends View {
```

Too Much Code

MyFragment.kt

fragment_my.xml

Some stuff in attrs.xml, styles.xml, more...

Goals

Unbundle from platform releases

Fewer technology **stack** flowcharts

Clarify state **Ownership** and event handling

Write **less** code

Flashback kuy, gan!

```
fun main() {
    println(“Isfa Ganteng!”)
}
```

Flashback kuy, gan!

```
fun Widgetkoe() {
    Text(“Isfa Ganteng!”)
}
```

Compose experimental

Inspired by React, Litho, Vue.js, Flutter

Consise and Idiomatic

Inherits the kotlin benefits

Compatible

With existing views

Declarative

Write UI using DSL

Accelerate Development

Less code and live preview

UI as a function

- Take data as input
- Emit UI hierarchy when invoked

```
@Composable
fun Widgetkoe(nama: String) {
    Text("$nama Ganteng!")
}
```


```
@Composable
fun Widgetkoe(nama: String) {
    Text("$nama Ganteng!")
}
```

```
class MyActivity: AppCompatActivity() {
      override fun onCreate(savedInstanceState: Bundle) {

          super.onCreate(savedInstanceState)
          setContent {
              Widgetkoe("Saya")
          }
      }
}
```

Composable building blocks

Composable functions are defined in terms of other composable functions

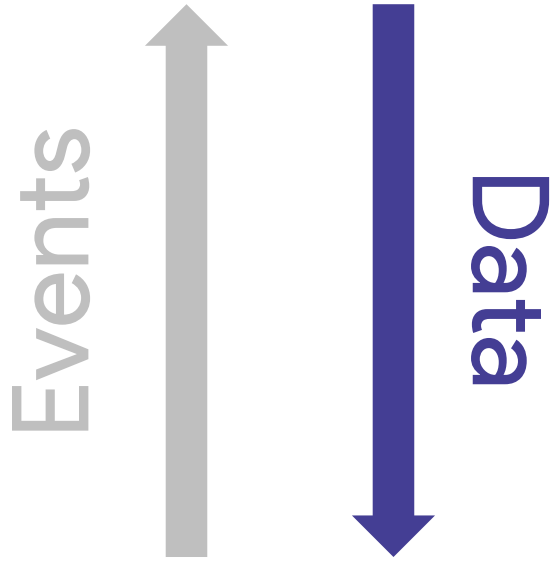
```
@Composable
fun NewsFeed(news: List<Data>) {
    ScrollingList(news) { data ->
        NewsWidget(data)
    }
}

@Composable
fun NewsWidget(data: Data) {
    Padding(8.dp) {
        Column {
            Title(data.title)
            Image(data.posterUrl)
        }
    }
}
```

```
@Composable
fun NewsFeed(news: LiveData<List<Data>>) {
    ScrollingList(news.observe()) { data ->
        NewsWidget(data)
    }
}
```

Top-down Data Flow

Important to understand how Data flows through your Application.




```
@Composable
fun NewsFeed(news: List<Data>) {
    ScrollingList(news) { data ->
        NewsWidget(data)
    }
}

@Composable
fun NewsWidget(data: Data) {
    Padding(8.dp) {
        Column {
            Title(data.title)
            Image(data.posterUrl)
        }
    }
}
```

```
@Composable
fun NewsFeed(news: List<Data>, onSelected: (Data) -> Unit) {
    ScrollingList(news) { data ->
        NewsWidget(data, onClick = { onSelected(data) })
    }
}

@Composable
fun NewsWidget(data: Data, onClick: () -> Unit) {
    Clickable(onClick) {
        Padding(8.dp) {
            Column {
                Title(data.title)
                Image(data.posterUrl)
            }
        }
    }
}
```

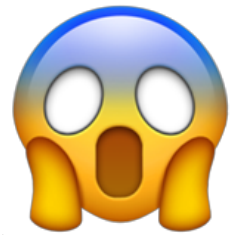
Recap

A UI API and component set for Jetpack

Composable function scale across layer of abstraction

First-class, one-way observable data flow

Less code, kept in one place



Ustt, how about View Compatibility?

```
@Composable
@GenerateView
fun Widgetkoe(nama: String) { ... }

<WidgetkoeView
      android:id="@+id/txt_widgetkoe"
      app:nama="@string/isfa_ganteng"/>


val widgetkoe: WidgetkoeView = findViewById(R.id.txt_widgetkoe)
widgetkoe.setNama("Isfa ganteng kan, gan?")
```

Let’s getting started

$ git clone https://github.com/andriiginting/jetpack-compose-experiment.git

$ ./jetpack-compose.sh

$ chmod +x jetpack-compose

experimental

It is good for production?

Absolutely, No.

Thank you!

Hiring!

Software Engineer - Android

drop your CV, here.
isfhani.ghiath@tokopedia.com