Public Channels

- # android-studio
- # dev-summit-2019
- # discussions
- # gson
- # introductions
- # java
- # learn
- # retrofit

Private Channels

Direct Messages

Group Direct Messages

Rohan Sharma 2019-11-02 10:58:23

@Rohan Sharma has joined the channel

Anuj Middha 2019-11-02 12:00:10

Welcome everyone!

:spock-hand: Abhishek Bansal, Tarun Yekbote

Anuj Middha 2019-11-02 12:02:45

while people are assembling, what was your favourite talk out of the 5 @Abhishek Bansal

Abhishek Bansal 2019-11-02 12:09:58

Thread Reply: My favourite one was I think Jetpack Compose. It cleared a lot of dust for me.

Anuj Middha 2019-11-02 12:11:06

Thread Reply: Yeah me too. Though I had no idea about Compose before this. I also liked the DI talk, especially the stuff that's coming in Dagger in the

Abhishek Bansal 2019-11-02 12:11:53

Thread Reply: Still can't wrap my head around Dagger. Koin for me for now 😕

Abhishek Bansal 2019-11-02 12:12:15

Thread Reply: would like to read up a bit more about Service Locators though

Anupam singh 2019-11-02 12:02:58

This is a serious online meetup else i would have posted,
"Aao aao" gif of sharukh khan from DDLJ

🤣 Shahbaz Hussain, Abhishek Bansal

Anuj Middha 2019-11-02 12:03:18

😂

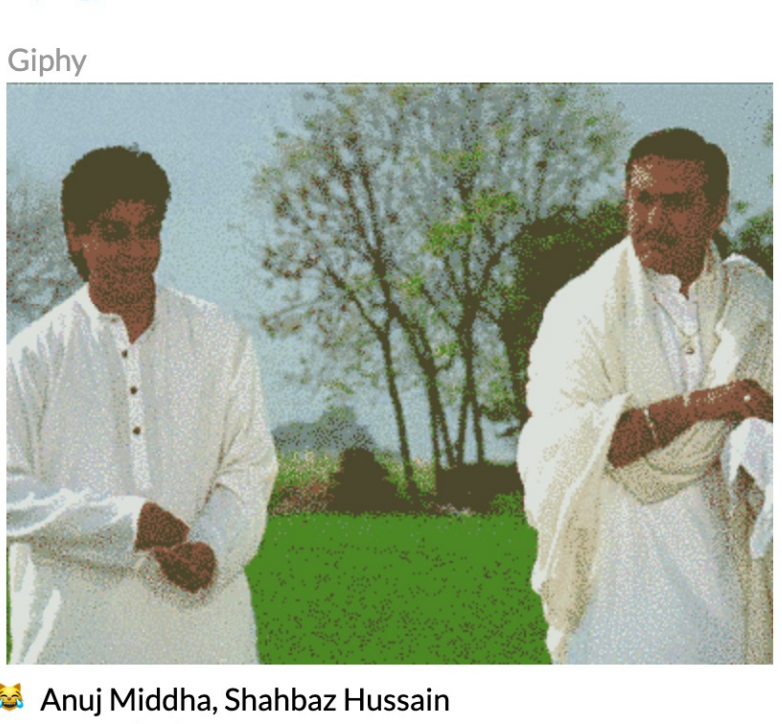
Anuj Middha 2019-11-02 12:03:47

not so sure about the `serious` part

Anupam singh 2019-11-02 12:04:44

http://gph.is/2cxuWoE

Giphy



😹 Anuj Middha, Shahbaz Hussain

Anupam singh 2019-11-02 12:06:10

@Anuj Middha lets go talk by talk in 5 threads, so that people joining in have context and history to follow.

Anuj Middha 2019-11-02 12:06:33

Sounds good.

Anupam singh 2019-11-02 12:06:54

okay i am reposting the videos one by one with summaries written by our team at UC

✔ Shahbaz Hussain, Tarun Yekbote, Abhishek Bansal, Annsh Singh

Anuj Middha 2019-11-02 12:46:08

Thread Reply: Did you guys divide up the talks and write summaries for all of them?

Anupam singh 2019-11-02 12:48:46

Thread Reply: yes

Anuj Middha 2019-11-02 12:50:17

Thread Reply: wow that's a great idea

Anupam singh 2019-11-02 12:08:42

Thread 1 : Fragments Past and future

https://www.youtube.com/watch?v=RS1IACnZLy4

YouTube
Android Developers
Fragments: Past, Present, and Future (Android Dev Summit '19)

Anupam singh 2019-11-02 12:09:09

Thread Reply: Target Audience : All Android Developers

`Past`

Fragment came out as Micro Activities. Everything an activity can do, fragment can do too. This meant lot of API in Activity was inherent in Fragment. A
activity is also sent to Fragment like onActivityResult, permission, picture-in-picture,multi-window.
What is our goal of Fragment?
Move to a world where Fragments have a focused API surface with predictable, sane behaviour without breaking compatibility.

`Present`

What are we doing to get there?
1, A good sane API needs to be testable. Introducing **FragmentScenario** for testing one Fragment in isolation.

- built on top of ActivityScenario
- onFragment() allows performing actions on a single Fragment
- can do scenario.moveToState(State.CREATED) or scenario.recreate

1. We had multiple ways of instantiating fragments : recreate on config change, FragmentTransaction add/replace, FragmentScenario We want one w
FragmentFactory.
    - It supports constructor injection
    - No requirement for a no argument constructor
    - Only one function to override - instantiate(classLoader, className). Given classname, up to you to load class.
    - Shorter way :

~~~
supportFragmentManager.commit{ add&lt;MyFragment&gt;(R.id.container) }
``` - You can use custom factory with FragmentScenario as :
~~~

val scenario = launchFragmentInContainer<MyFragment>(factory=MockFactory())
```

1. There was inconsistency in the ways fragments are added with tags and through FragmentTransaction. To provide consistent behaviour, introducin
    - It extends and aims to replace FrameLayout.
    - Only allows Fragment views
    - Replaces <fragment> tag when you add class or android name attr
    - helps with z-ordering in Fragment Animations
2. One way to handle Back Press : added **OnBackPressDispatcher**. It lets any lifecycle owner listen for back
3. Leaning into Arch Components
    - ViewModel : easy to get instance of ViewModel
    - Lifecycle : can now use regular lifecycle methods when adding Fragment to ViewPager or one adapter. Only current fragment will be resume

`Future`

- Multiple Back Stacks : allow saving and restoring whole stack of Fragment without losing state
- Returning results : Its not easy to talk between Fragments and with Activity right now. Target Fragment is considered harmful. onActivityResult fo
better API for startActivityForResult and pass result to another fragment
- Fragment has multiple lifecycles : one of itself, one of its view. Because of backstack, view may be destroyed but fragment lives on. Need to be same
fragment gets destroyed. When view recreated, fragment too.

Hitanshu Dhawan 2019-11-02 12:14:29

Thread Reply: Fragments: Constructor Injection 😍

✔ Abhishek Bansal, Tarun Yekbote, Shahbaz Hussain

Anuj Middha 2019-11-02 12:26:24

Thread Reply: That, and the FragmentScenario were the best parts from this talk

Abhishek Bansal 2019-11-02 12:33:03

Thread Reply: Multiple Backstacks would be useful as well for apps with Bottom Nav

👍 Annsh Singh, Shahbaz Hussain

Anuj Middha 2019-11-02 12:34:15

Thread Reply: Doesn't feel natural to me for some reason. Maybe because we've never had it so..

Abhishek Bansal 2019-11-02 12:37:39

Thread Reply: Yeah so we implemented it on our own when we were doing that `XPrize` thing if you remember. That app had bottom nav and its difficul
when you switch tabs in ViewPager. This should solve that problem

👍 Anuj Middha, Shahbaz Hussain

Anupam singh 2019-11-02 12:10:05

Thread 2 :
An Opinionated Guide to Dependency Injection on Android

https://www.youtube.com/watch?v=o-ins1nvbDg

YouTube
Android Developers
An Opinionated Guide to Dependency Injection on Android (Android Dev Summit '19)

Anupam singh 2019-11-02 12:10:24

Thread Reply: DI: Dependencies are provided to a class instead of the class creating them itself.

Without DI

~~~
class Car {
    private val engine: Engine = Engine()
}
```With DI
~~~

class Car(private val engine: Engine) {

}
```

Importance of DI

- Reusability of code
- Good balance of loosely-coupled dependencies
- Ease of refactoring
- Ease of testing

Choosing right DI solution based on Project size

- Small: Manual DI, Service Locator, Dagger
- Medium: Dagger
- Large: Dagger

[NEW] [COMING SOON] Dagger with Kotlin and Android

- No need of `@JvmStatic` in `@Module`
- No need of `@field:Username` just use `@Username`
- No need of `.inject()` method in Activity

Hitanshu Dhawan 2019-11-02 12:11:10

Thread Reply: I liked that Google is getting serious about DI and making Dagger easier in android.
Dagger + Kotlin was and still is a mess. It's because the annotation processors are not so interoperable out of the box.
This is more explained in this talk
https://www.youtube.com/watch?v=a2RoLFzrFG0
They also released a tutorial/training blog : https://developer.android.com/training/dependency-injection

Anuj Middha 2019-11-02 12:12:27

Thread Reply: I am also super happy about the proposed changes in Dagger. I currently use Koin in my projects, but will for sure switch to dagger whe

Anuj Middha 2019-11-02 12:13:40

Thread Reply: Though I do not necessarily agree with the Service Locators not scaling with the app size part. I think Koin has done a good job

Abhishek Bansal 2019-11-02 12:15:25

Thread Reply: Why do you want to switch? Any particular reason or its just that Google's heavy support?

Annsh Singh 2019-11-02 12:18:37

Thread Reply: I haven't seen the latest changes but I saw a lot of backlash for the same on Twitter.

Anuj Middha 2019-11-02 12:19:31

Thread Reply: Static code generation is always better - One the most valid points I felt justified Dagger

Hitanshu Dhawan 2019-11-02 12:20:02

Thread Reply: @Anuj Middha I've not used Koin in my projects.
But AFAIK,
Dagger is DI and Koin is Service Locator.
In DI we pass the dependencies and in Service Locator we get them from the SL.
So the SL way becomes a little tricky to unit test as we can't pass the dependencies.
@Anuj Middha Have you faced any issue with unit testing with Koin?

Anuj Middha 2019-11-02 12:20:53

Thread Reply: You just instantiate Koin with different modules for your tests. It works nicely

Abhishek Bansal 2019-11-02 12:21:46

Thread Reply: Unit testing is not a problem with Koin. They have their testing libraries which makes it super easy. ViewModels are constructor injecte
dependencies with mock ones.

Anuj Middha 2019-11-02 12:22:39

Thread Reply: Having more number of smaller modules helps, as you can swap them out with the mock ones

Annsh Singh 2019-11-02 12:23:14

Thread Reply: It took me 20mins to understand Koin
And a few days with Dagger

Anuj Middha 2019-11-02 12:23:55

Thread Reply: With the new proposed features of Dagger though I feel its gonna be as simple to implement as Koin

Anuj Middha 2019-11-02 12:24:12

Thread Reply: and the Integration with Architecture components looks really handy

Hitanshu Dhawan 2019-11-02 12:26:18

Thread Reply: Ohh, nice. Koin looks promising.
Will surely have a look at it.
Thanks @Anuj Middha and @Abhishek Bansal

Annsh Singh 2019-11-02 12:27:08

Thread Reply: I would like you to check out this thread about Dagger - https://twitter.com/VasiliyZukanov/status/1188549948037570561?s=19

Anuj Middha 2019-11-02 12:27:31

Thread Reply: We have koin in a multi module project, and our core modules are actually shared among multiple apps. Koin was really easy to setup, a

Anuj Middha 2019-11-02 12:33:17

Thread Reply: @Annsh Singh the twitter thread you linked to goes on to blast the architecture choices of the code lab. While at the beginning of the co
probably not the best way to write your android app. They are only trying to showcase the features of Dagger in this.

➕ Abhishek Bansal

Annsh Singh 2019-11-02 12:36:34

Thread Reply: @Anuj Middha The main pain point of Dagger I believe is that nobody really knows how to perfectly implement it.
I have seen so many articles/videos on it and everyone is doing one thing or the other which creates doubts in my mind as to what the proper implementa
I don't know if it's just me, but this was my first impression of dagger.

👍 Anupam singh, Himanshu Khati

Anuj Middha 2019-11-02 12:38:06

Thread Reply: Agreed. That is what makes it easier with Google coming out with standard components and such.. Its best if they are "opinionated"

👍 Hitanshu Dhawan, Annsh Singh

Abhishek Bansal 2019-11-02 12:38:46

Thread Reply: @Annsh Singh I agree I once wrote one long article on Dagger myself and at this point I have no clue 😕

🤣 Anupam singh, Annsh Singh

Annsh Singh 2019-11-02 12:43:36

Thread Reply: A talk on Dagger as well please @Anuj Middha 😅

➕ Hitanshu Dhawan, Ayusch Jain

Anuj Middha 2019-11-02 12:44:22

Thread Reply: haha. One the new feature are out perhaps?

➕ Abhishek Bansal

Annsh Singh 2019-11-02 12:54:22

Thread Reply: @Anuj Middha Whenever you feel comfortable, just help us Developers find something to follow. A fixed pattern that works and is scala

👍 Anuj Middha

Anupam singh 2019-11-02 13:05:53

Thread Reply: @Anuj Middha personal question, how come you are so invested in tech, people with your background and position generally dont find

In short what is fuelling you, we will all like to take a sneak peak in that insanely hardworking brain and humble demeanour

👏 Hitanshu Dhawan, Annsh Singh, Abhishek Bansal, Ayusch Jain

Anuj Middha 2019-11-02 13:09:48

Thread Reply: Those are too kind words good sir, not fully deserving perhaps. Good for me that I recently took up a position more focused on Tech, lea
capable people 🙂

👍 Abhishek Bansal, Anupam singh

Anupam singh 2019-11-02 12:11:17

Thread 3 : LiveData with Coroutines and Flow https://www.youtube.com/watch?v=B8ppnjGPAGE

YouTube
Android Developers
LiveData with Coroutines and Flow (Android Dev Summit '19)

Anupam singh 2019-11-02 12:12:22

Thread Reply: We have multiple scopes and structuring the concurrency for scopes is a difficult task like cancellation of a task when view destroyed e
Structured concurrency is referred to managing jobs or can say garbage collector for jobs. If doesn't need a job, it call automatically. In jetpack, provide s
use different kinds of scopes according to our requirement like lifecycle scope (which get cancelled if the lifecycle is destroyed), view model scope etc. U
important.

Since lifecycle 2.0, using name builder called liveData. This liveData builder gives a coroutine scope which is like a scope to your liveData(start executing
cancelled when live data is not used). Emit() can be called from this scope. One new function is emitsource(), which receives a live data. Other livedata di

All suspending functions in kotlinx coroutine are cancellable. If not calling any cancellable functions then its fine now. Cancellation is cooperative, need t
active using isActive property.

OneShot operation (update multiple transformations)
There is a new API called flow. Flow is part of kotlin coroutines. In flow, we need to convert the Flow to livedata somehow, so livedata coroutine builder
any flow to livedata. From this we can remove multiple handling and boiler plate code of livedata.
In livedata, Using switchmap instead of map when transformation need to be done on multiple items which you are receiving or observing. In flow, just ca
Retrofit support coroutine function since 2.6.0
Room support coroutine function since 2.1.0

suspendCancellableCoroutine{} -> this is basically an adapter between the coroutine world and the callback based world. Like make an API request, if AP
continuation, similarty for failure callbacks. If coroutine is cancelled, API request is also cancelled. Custom callbacks can also be created and handled usi
callback, flow is close, so cannot return anything. For that need to use awaitClose(), which suspends until the flow is not collecting any more data , in that

Anuj Middha 2019-11-02 12:15:58

Thread Reply: Their coroutine code looked so familiar, because I use the exact same patterns in my code. It was so nice to see the liveData coroutine b
write

Anuj Middha 2019-11-02 12:16:53

Thread Reply: And anyone nostalgic about RxJava will be super happy with Flow 🙂

Abhishek Bansal 2019-11-02 12:19:15

Thread Reply: RxJava still Rules when it comes to operators like `debounce` or `throttle` or any windowing and timing related operations. These are the
implement otherwise.

Anupam singh 2019-11-02 12:13:23

Thread 4 :What's New in Jetpack Compose

https://www.youtube.com/watch?v=dtm2h-_sNDQ

YouTube
Android Developers
What's New in Jetpack Compose (Android Dev Summit '19)

Anupam singh 2019-11-02 12:14:23

Thread Reply: Jetpack Compose is a declarative way of writing Views

- to make development easy
- to make apis that just do one work, e.g. TextView doesn't know about padding/margin

It is done by adding @Composable to methods to make views.

[NEW] Material + Animations
[NEW] Tools e.g. visual editor (@Preview)
[NEW] Kotlin first

Android team is working on adding Constraint Layout in future.
Android team is creating Material Components

We can also make Custom Views.
In Draw method we will get canvas and using that canvas we can make our custom views.
In Layout we can make our custom layouts

[NEW] Developer Preview is out (0.1.0-dev02)

Jetnews is a sample news reading app, built with Jetpack Compose.
https://github.com/android/compose-samples/tree/master/JetNews

State Management is done by adding @Model annotation on top of the class that you want to observe.
@Model works even if we change nested data (Not possible in LiveData).

Coming Soon

- View Compatibility (Interoperability)
- View Binding
- Backward Compatibility

d.android.com/jetpackcompose

Jetpack Compose available on Android Studio 4.0 Canary 1

GitHub
android/compose-samples
Contribute to android/compose-samples development by creating an account on GitHub.

Anuj Middha 2019-11-02 12:17:49

Thread Reply: This was my favourite talk. Compose really seems to talk a lot of the complexity out of managing your view and data consistency, as if by

Anuj Middha 2019-11-02 12:18:07

Thread Reply: I am planning to try it out this weekend only

Anupam singh 2019-11-02 12:18:54

Thread Reply: @Hitanshu Dhawan has some reservations about it in terms of simple structuring in code he is our expert will add more

Anupam singh 2019-11-02 12:19:19

Thread Reply: what about everyone following the declarative ui format now

Annsh Singh 2019-11-02 12:20:54

Thread Reply: Yup even iOS is jumping on the Declarative UI format train and I am sure eventually all Developers will be switching

Abhishek Bansal 2019-11-02 12:24:46

Thread Reply: It will change the whole paradigm I think. We will have to rethink the app architecture wise. Bi-furcating ViewModel into smaller compo

Hitanshu Dhawan 2019-11-02 12:34:40

Thread Reply: Okay, so may be I'm the only one here not liking Compose 😜
But it's completely new to me and yes making Composables for simple linear layouts, frame layouts is fine, but how will it scale to more complex layouts l
(coming soon).
Also, for complex views it will be deeply nested and maybe difficult to maintain.

Anuj Middha 2019-11-02 12:35:14

Thread Reply: Components?

Hitanshu Dhawan 2019-11-02 12:36:15

Thread Reply: I like the fact that they are separating the responsibilities. e.g. TextView will only show text and Padding will only add padding.

Anuj Middha 2019-11-02 12:36:44

Thread Reply: You can "compose" more complex functions (views) from smaller ones no?

Annsh Singh 2019-11-02 12:40:43

Thread Reply: @Hitanshu Dhawan I would suggest you to try flutter to really feel the difference. UI in code is so easy to manage. Also, I was surprised
"visualize" UI in my mind while building it in code without any visual feedback in a preview screen.

Abhishek Bansal 2019-11-02 12:40:47

Thread Reply: Thats something we have to see, I have heard people complaining about this nesting in Flutter as well.

Annsh Singh 2019-11-02 12:42:24

Thread Reply: @Abhishek Bansal Nesting will bug you for a while but the team has setup some great visual cues to help you understand the start and e
get used to it once you build a small app.

Anuj Middha 2019-11-02 12:43:02

Thread Reply: @Annsh Singh have you built any complex UIs in code?

Abhishek Bansal 2019-11-02 12:43:08

Thread Reply: Okay I guess I will have to try it out first.

Anupam singh 2019-11-02 12:43:24

Thread Reply: @Annsh Singh okay but what about new devs who preferred android also because to get started it was really easy to create UI's just dr

Do you think composing ui might act as a deterrent in future for new devs?

Anuj Middha 2019-11-02 12:44:03

Thread Reply: Does anyone actually use the drag and drop feature? 🤔

➕ Abhishek Bansal

Hitanshu Dhawan 2019-11-02 12:45:56

Thread Reply: Also I'm unable to visualize how will the code look when compose has constraint layouts.
Constraints, Chaining, Circular Positioning. IMO the code will become very hard to understand in Declarative pattern.
Help me understand more if I'm missing something.

Abhishek Bansal 2019-11-02 12:46:52

Thread Reply: Well we have to wait for it I guess, that list filter example was pretty sick though!

Anuj Middha 2019-11-02 12:47:50

Thread Reply: I also love the ability to preview lists and views with dummy data in the preview. Helps in adjusting the paddings, margins etc.

Annsh Singh 2019-11-02 12:48:03

Thread Reply: @Anuj Middha Not that complex. But I have worked with a lot of widgets in Flutter and was quite impressed with the ease of implemen

👍 Anuj Middha

Abhishek Bansal 2019-11-02 12:48:24

Thread Reply: I have only seen absolute beginners using drag and drop, like college students. They build app for just one device.

Anuj Middha 2019-11-02 12:49:33

Thread Reply: Something like this @Abhishek Bansal? Top margin: 129 px, width: 423px... 😜

😲 Hitanshu Dhawan

Abhishek Bansal 2019-11-02 12:49:45

Thread Reply: yeah

Abhishek Bansal 2019-11-02 12:51:53

Thread Reply: When I started with `ConstraintLayout` I started with drag and drop because Google heavily promotes it that way. But now `XML` is way fa

✔ Hitanshu Dhawan

Annsh Singh 2019-11-02 12:52:15

Thread Reply: @Anupam singh On the contrary, a lot of students are trying out flutter and loving it. Just because of the fact that not a lot needs to be c
With the Declarative UI coming soon in Android, I think it'll be easy for students to switch to Android from Flutter.

I think people like options. Especially the ones starting out. It'll be fun to see what they pick, the good old xml or the jetpack compose, or maybe a mix of l

Anuj Middha 2019-11-02 12:53:54

Thread Reply: the best part is that they are planning both to be compatible. so you do not have to choose one over the other

✔ Hitanshu Dhawan, Abhishek Bansal

Abhishek Bansal 2019-11-02 12:54:17

Thread Reply: Not only students but there is a good commercial market for tools like `Flutter` and `RN` so who knows what future holds.

Anuj Middha 2019-11-02 12:55:55

Thread Reply: I am definitely trying out the beta soon

Annsh Singh 2019-11-02 12:56:02

Thread Reply: I just wish Animations get simpler with Compose.

Anupam singh 2019-11-02 12:57:19

Thread Reply: @Annsh Singh fair points,

But animations will take time in IMO, simpler maybe yes

Annsh Singh 2019-11-02 12:58:55

Thread Reply: @Anupam singh True, still hoping for the best :)

Anupam singh 2019-11-02 12:14:57

Thread 5 : https://www.youtube.com/watch?v=jCmJWOkjbM0

YouTube
Android Developers
Coding in Style: Static Analysis with Custom Lint Rules (Android Dev Summit '19)

Anupam singh 2019-11-02 12:15:23

Thread Reply: Summary:

Audience - Android Tooling devs

Agenda- How to write your own lint rules

Create a Java libray module and in which lint rules will be defined.
Create an IssueRegistry i.e. A list of issues that this lint library will contain.
Create issues and with each issue refer a Detector. Main code will go inside the detector.

The detector is where the Magic happens. It can be used to parse an XML/java/kotlin file.
With each detector you tell what to detect and once that is found it returns a callback where you can check if the implementation is good or bad.

One example is if a file is deprecated in source code. A lint can be created such that if a dev tries to use that class it gives a compilation error.

Lint also gives option to create LintFix. This is the same as which we use in general practice (Option + return) to fix lints. Here you can give how a particu

They briefly mention something aclled UAST and PSI which I didnt understand much. In my limited understanding those are used to access the code bloc
navigate upwards or downwards in the tree. One example in kotlin is when you already check null once above and in the subsequent code you use ? it giv

Abhishek Bansal 2019-11-02 12:32:01

Thread Reply: Well nobody is talking about this 😜 . This was a good primer though. What are some example lint checks that you would like to have for
use your custom extension methods instead of Android boiler plate?

Hitanshu Dhawan 2019-11-02 12:46:48

Thread Reply: Yes, I really liked the example of `<a href="http://Log.wtf">Log.wtf</a>()` 🤣

😜 Abhishek Bansal, Anupam singh

Hitanshu Dhawan 2019-11-02 12:49:30

Thread Reply: Even, I was thinking of making lints for making consistent coding practice across our team.
Just like you mentioned.
e.g. when using `Color.parseColor()` we can have a lint which will tell the developer to use UC specific Color class and it will change it to `UcColor.parseCo`
color in a better way.

Abhishek Bansal 2019-11-02 12:56:36

Thread Reply: Yeah this helps a lot and also makes code review process faster.

Ankur Khandelwal 2019-11-02 12:20:34

@Ankur Khandelwal has joined the channel

Anuj Middha 2019-11-02 12:39:34

Following 5 separate threads is tough..

Anupam singh 2019-11-02 12:43:54

Thread Reply: but otherwise it will be all choas.

➕ Abhishek Bansal

Anupam singh 2019-11-02 12:44:17

Thread Reply: and last think you want in a android discussion is more

"Fragmentation"

🤯 Hitanshu Dhawan, Abhishek Bansal, Annsh Singh

Anuj Middha 2019-11-02 12:44:40

Thread Reply: :man-facepalming:

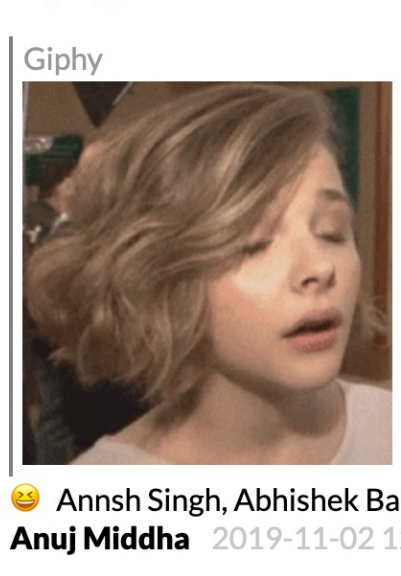
Anuj Middha 2019-11-02 12:44:44

Thread Reply: haha

Anuj Middha 2019-11-02 12:42:04

http://gph.is/1a9P4YU

Giphy



😆 Annsh Singh, Abhishek Bansal, Anupam singh, Hitanshu Dhawan, Shahbaz Hussain, Puneet Verma

Anuj Middha 2019-11-02 12:51:02

Overall I think I am more excited about the things to come than what is already there..

Anupam singh 2019-11-02 12:51:51

Thread Reply: Indeed

Anuj Middha 2019-11-02 12:52:39

Thread Reply: Perhaps because we did not choose any of the "What's new in" talks 🤔

Anupam singh 2019-11-02 12:57:56

Thread Reply: Haha yes

Anupam singh 2019-11-02 12:58:49

Thread Reply: we all learn slowly, will plan better videos , better talk construct in future

Plus people are always more excited by future

Hitanshu Dhawan 2019-11-02 12:51:28

Haha, correct 😜

Anuj Middha 2019-11-02 12:52:04

Flow, Changes in Dagger and Fragments, Compose

Anupam singh 2019-11-02 13:01:41

Thread Reply: okay with few people joining in, i guess discussion will saturate in next 15 mins.

lets close the chat hour,

and use this thread for people to post videos from the session, and discuss in async fashion

Uptil kotlinconf

Sort of continuous discussion, async though

Anuj Middha 2019-11-02 13:02:46

Thread Reply: Yup.

Anuj Middha 2019-11-02 13:03:45

@channel Thank you all for some lovely conversations. We'll use this channel for any follow up discussions on all tech talks.

👍 Abhishek Bansal, Annsh Singh, Anupam singh, Shahbaz Hussain

Anuj Middha 2019-11-02 13:05:32

Sayonara and a great weekend to everyone!

Annsh Singh 2019-11-02 13:05:44

This was a nice idea 👌 :skin-tone-2:
Thanks for organizing @Anuj Middha @Abhishek Bansal @Anupam singh
Also, I must point out that all the summaries were well structured @Anupam singh 👍 :skin-tone-2:

Abhishek Bansal 2019-11-02 13:06:03

Yeah, special thanks @Anupam singh

Anupam singh 2019-11-02 13:07:39

No it was my team at UC, i did only 5%,

we will post a very long article on all relevant videos, with nested links, summaries of all the talks by next saturday.

yes it will be TLDR , but worth investing time in maybe one summary at a time

👍 Abhishek Bansal, Annsh Singh, Shahbaz Hussain, Niharika Arora

Annsh Singh 2019-11-02 13:09:15

Thread Reply: They put in a lot of effort into this. Kudos to your team :)

Anupam singh 2019-11-02 13:09:49

Thread Reply: Well they have no choice,

Mid year Performance review is coming 😂

🤣 Abhishek Bansal, Annsh Singh, Shahbaz Hussain, Ayusch Jain

Annsh Singh 2019-11-02 13:13:34

Thread Reply: Hahaha 🤣

Anupam singh 2019-11-02 13:09:01

Happy Weekend guys.
~~~