

What's New in Jetpack

1. Overview

- a. Suites of libraries to make developers build app easily
- b. Since Google launched the Jetpack, the adoption rate has increased up to 47 percent (surveyed on top 1000 apps, at least one or two libraries of jetpack has been used)



- c. We see the changed icon!

2. Hilt

- a. Dependency Injection Library built on the Dagger
- b. Why DI?
 - i. Greater code reusability due to decoupling between components.
 - ii. And ease of testing
 - iii. 49 % of developers asked for a DI solution
- c. Why Dagger?
 - i. 74% of top 10K apps adopted Dagger.
 - ii. Steep learning curve -> background of Hilt.
- d. Hilt
 - i. Designed for Android, it knows Android components like ViewModel and its scope
 - ii. Provides new annotations, and pre-defined scope especially about Android
 - iii. Installing a Dagger Module is also available. Hilt will discover it automatically.
- e. Hilt in Actions - e.g Google I/O app. Google team saw the great amount has been reduced from DI code by using Hilt. (75%)
- f. Integrated with Jetpack(ViewModel, Fragment, and WorkManager), Scopes for the Android, Box of test Apis, and android studio integration
- g. Alpha stage

3. App Startup -

- a. Faster application initialization, to reduce negative effect on launch performance
- b. Content Provider to initialize automatically, including jetpack libraries like workmanager and lifecycle.
- c. Provides a straightforward and performant way.
- d. Single ContentProvider shared between all these initializers reducing the application startup time.
- e. Automatically added Trace Point to figure out the performance.

4. Android Game SDK
 - a. Gaming support now in Jetpack!
 - b. Two important modules in Gaming SDK
 - i. Frame pacing API - to maintain a steady frame rate and lower input latency (detect the expected frame rate and auto-adjust frame presentation times)
 - ii. Performance Tuner - performance in Android Vital
5. Benchmark 1.1
 - a. Support for CPU profiling benchmarks
 - b. Memory allocation developer makes
6. Paging 3
 - a. Kotlin / Coroutines and Flow
 - b. Headers, Footers, and Separator
 - c. Loading State and Retry
 - d. Compatible with Paging 2
 - e. PagingSource, Pager, and Presenting
 - f. Alpha, and First Kotlin-based library
7. CameraX
 - a. Beta Stage, focusing on reliability and documentation
 - b. Runs on 400M Devices in use.
 - c. PreviewView
 - i. camera preview, interactions with lifecycle.
 - ii. Less buffering, and better power efficiency
 - d. Yuv to RGB conversation to do image analysis
8. WorkManager
 - a. Allowing us to run deferrable background job
 - b. Not relying on Jobscheduler, but in-process scheduler
 - c. Now supports Delayed workers and periodic work requests
 - d. No longer imposes scheduling limits -> improves throughput of work requests.
 - e. Supporters long-running work that should be kept alive by OS. <- Using Foreground Service. (more than 10 minutes. But have to show notification.)
 - f. Diagnostics
 - i. New Diagnostics Api that we can invoke with ADB.
 - ii. Can dump diagnostics to logcat
9. Navigation - allows us to navigate between different screens of the app easily
 - a. Navigation 2.3
 - i. Supports Dynamic feature modules, with corresponding classes annotated.
 - ii. Deep Linking Improvement - by using one parameter in the graph.
 - iii. Returning Result
 1. Each screen in the app has a NavBackStackEntry.
 2. It gives us access to the same state of that entry as well.

3. Navigation uses the `SavedStateHandle` class to pass data between screens, to ensure that results are kept even over configuration changes
4. Our fragment can access the previous fragment `SavedStateHandle` by using the previous `BackStackEntry`.
5. Once we obtain the `saveStateHandle` from the previous entry, we can set the result values on the `SavedState`.
6. To observe the result, we can get the same value from the `SavedState` of the current `BackStackEntry` and observe the value in `livedata`. It means observation is lifecycle aware.

10. Permissions and Activity

- a. `ActivityResultContracts` API - Activity 1.2.0 alpha2
 - i. Replacing `startActivityForResult`
 - ii. Type safe contracts for common intent
 - iii. Not only requesting permission (single or multiple), but also replacing some jobs using intent such as taking pictures, querying the contacts.

11. AppCompat

- a. Lint Rule
 - i. Library-specific Lint rule
 1. AS automatically shows the warning for users to replace the attribute to the one available in AppCompat. (in xml)
- b. AppCompat and Dark Mode
 - i. More reliable Dark Theme
 - ii. Configuration Override API for users to customize their theme
- c. Webkit and Dark Mode
 - i. Forced Dark mode support for `WebView` in 1.2.0.
 - ii. When enabled, the `Webview` will render size and Dark Theme are supported, or forcibly in more certain colors if the website doesn't support dark theme

12. And so on!