Reactive Android Intro: issues, examples and how to succeed with Rx and Cascade

Functional development styles are increasingly popular. Reactive development that links long-lived objects with pure functions is hot for good reason. We will quickly cover the basics and the most popular Android reactive library, Rx Java. We quickly go past the basics to discuss openly the real world issues you face when shifting to reactive designs. A special feature will be the first public announcement of Cascade, a new reactive library which focuses on the developer experience, debuggability and unleashing the raw performance of fully concurrent development.

Talk Outline, Still very much a draft

- Introduction to reactive and functional programming.
- Talk about observers and observables.
- Introduce Rx, talk about operators, create observables, threading, subjects, subscriptions (basic concepts)
    - Why is so big library?
- Show how this works with practical example: search box -> use operators like map, change threads, implement dynamic search with debounce. Goal: show how powerful this tool is and how it removes a lot of boilerplate compared to traditional ways of doing things.
- Talk about learning curve and risks! Show the risks and what could go wrong.
    - Leaks: Subscriptions hold references to observer and observable, easy to leak contexts -> enable strict mode.
    - Error handling: RxJava doesn't swallow/handle errors by default, it requires explicit error handling. If something goes wrong in the chain you can get an error log with no hooks to your code (all with Rx internals) -> Sometimes very difficult to track down -> Show tools Rx provides to handle errors (super quick)
    - Debugging: The source code is available (is open source) but the internals are not easy to follow.
    - Assumptions: It is very common and easy to assume how an operator works but it is important to know what it does internally, it can have unexpected side effects (unsubscribing, emitting too slow or too fast, etc.)
    - Breaking the chain: Makes spaghetti code -> Don't break the chain!
    - Back pressure: What if the source emits elements faster than the consumers can consume? CRASH -> Show very quickly tools Rx provides to handle this
    - Hot and cold observables: When does the observable start emitting items? Important to not lose items.
- Talk about other libraries that enable RxJava -> Retrofit.
- Introduce and world first announcement of an ongoing project led by Paul, an early stage similar reactive library for Android called "Reactive Cascade"
    - Similar but subtly different structure to Rx
    - Concurrent and parallel by default, speed gains with similar structure
    - Heavy use of lambdas (Retrolambda)
    - Focus on solving the common issues that come up in reactive architectures (debugging, memory leaks, work prioritization, concurrency, user experience)

- Side by side interactive example
    - Reactive MVVM mapping and filtering to a displayed list
    - Code styles shown and compared with traditional, dicussing development speed and maintenance
    - Runtime performance comparison
    - Thread model comparison
- Invitation to join the discussion