RxJava et Java 8
Speaker : **Simon Basle** – Zenika

L'enfer des callbacks
Les applications d'aujourd'hui son multi-threadées , asynchrones.
On a besoin d'un code qui reste lisible.

Comment rester lisible ?
Solutions :

1. L'utilisation des Future<T> de Java
2. Callback toute simple : le problème est qu'elles ne se composent pas et deviennent vites illisibles

Exemple avec 3 callbacks imbriqués, une énorme tabulation, et 2 slides de code => enfer des callbacks

Une solution partielle : RxJava, solution open source développée par Netflix. Vient de .NET. Pendant de Iterable / Iterator. Devient Observable / Observer, mais en mode « push ».
RxJava permet de composer des programmes asynchrones en utilisant des séquences observables.
Très flexible : gestion de liste et même de flux inifins.
Références :

- Reactive Manifesto.
- Initiative Reactive Streams

Interface Observer<T> avec 3 méthodes
Interface Observable permettant de s'abonner et de transformer (filter, skip, distinc), de combiner des flux observables (merge, zip, reduce).
La gestion des erreurs : onErrorReturn (valeur par défaut) et onErrorResumeNext

Avec RxJava avec le code initial reste un peu illisible : le code est tout de même mieux structuré.
Java 8 simplifie encore plus le code d'exemple : fonction de 1er ordre, lambdas
1 slide, 28 lignes de code, 1 imbrication, 4 tabulations

Avec Java 8 et RxJava, on va pouvoir faire de l'asynchrone sans callback et avec un minimum de pollution visuelle. Le code reste lisible et est compréhensif.