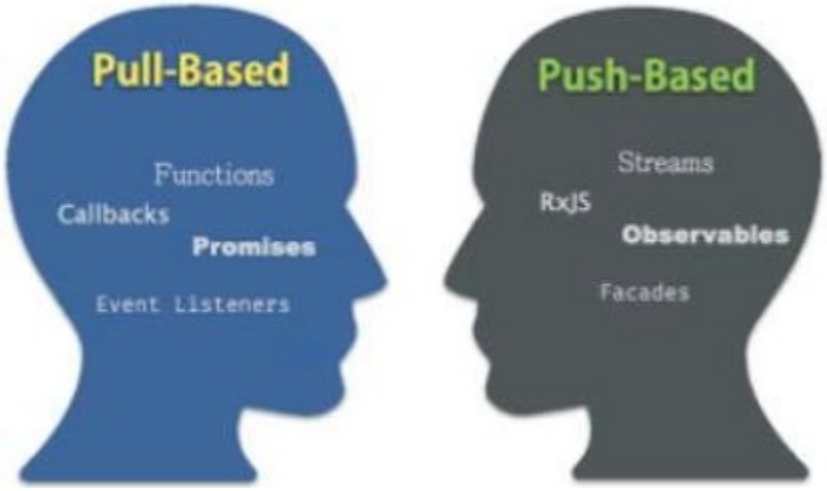
Pull based

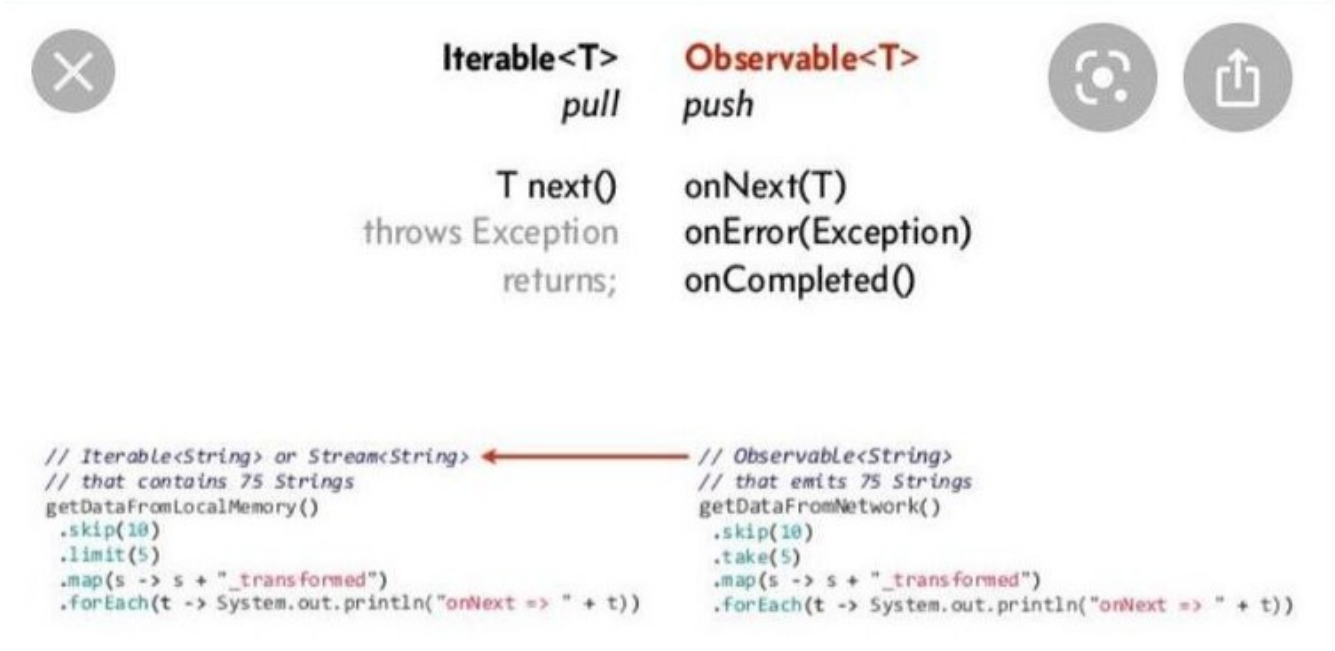
```
// I want some toast.  I will pull it out of the toaster when it is done.
// I will do nothing until the toast is available.
Toast t = toaster.MakeToast();
t.AddJam();
```

Push based

```
// I want some toast.  But it might take a while and I can do more work
// while I am waiting:
Task<Toast> task = toaster.MakeToastAsync();
Toast t = await task;
// await returns to the caller if the toast is not ready, and assigns
// a callback. When the toast is ready, the callback causes this method
// to start again from this point:
t.AddJam();
```

https://stackoverflow.com/questions/51254117/what-is-difference-between-push-based-and-pull-based-structures-like-ienumerable





https://www.uwanttolearn.com/android/pull-vs-push-imperative-vs-reactive-reactive-programming-android-rxjava2-hell-part2/

Imperative Approach:	Reactive Approach:
int val1 = 10;	int val1 = 10;
int val2 = 20;	int val2 = 20;
int sum = val1 + val2;	int sum = val1 + val2;
System.out.println(sum); // 30	System.out.println(sum); // 30
val1 = 5;	val1 = 5;
System.out.println(sum); // 30	System.out.println(sum); // 25