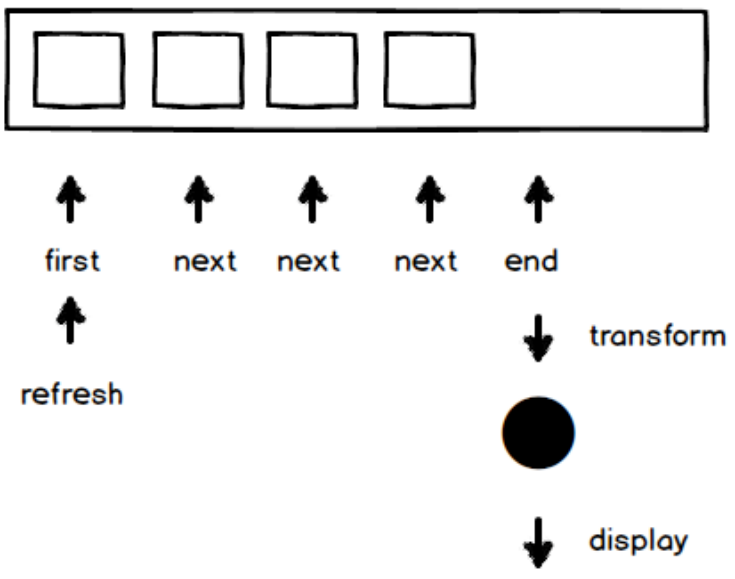
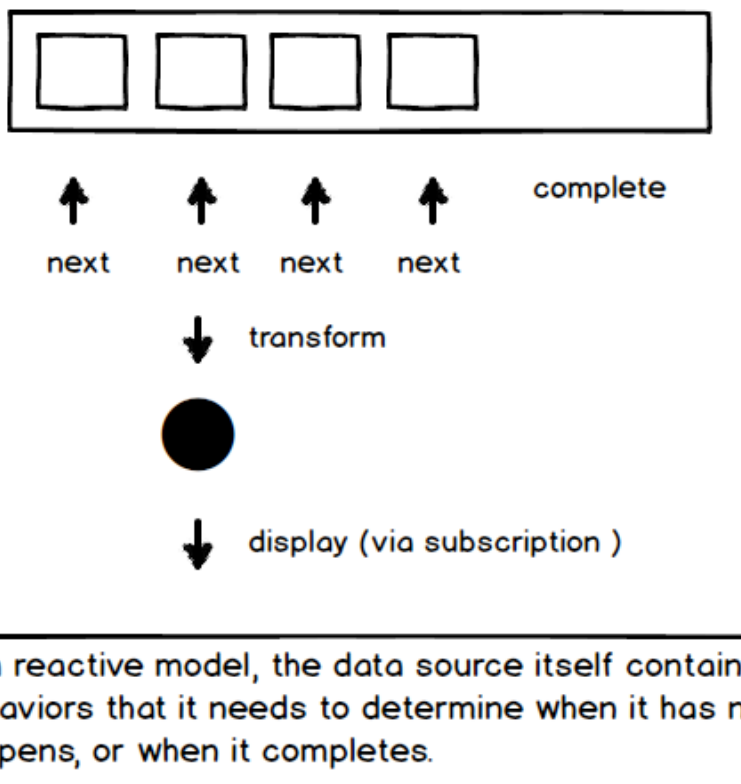
RxJs - Reactive Programming Library

Facebook Comments Array



Non-reactive model
Looping through an array
Reaching the end
Then utilizing that information

Facebook Comments Stream (Observable)



Reactive model
Receives streams of information
Perform individual transformations on streams
Over-time operation

In a reactive model, the data source itself contains all the concepts and behaviors that it needs to determine when it has new data, when an error happens, or when it completes.

- An Observable is simply a reactive data source. A reactive data source is a data source that produces data over a period of time - and at some point will either error or complete or possibly never complete until the page is closed.

- Whenever the observable gets data, it produces a result.
- You can attach as many pipelines of transformation, or subscriptions, to a particular observable. The pipeling could transform that new data, and then display it on a page.
- Once subscribed to an observable, whenever there is new data, all subscriber will receive that update.

- In the first example, we are pulling data from the model by iterating through it, getting the information, then doing something with that information.

- In the second example, the data is pushing to us without any further instructions, we then subscribe to that data, receive the new streams of updates as they come, and do something with that information.

- Uni-directional data flow where our data source are kind of the entry points to our application, and then all of our application logic depend on the data source as well.

- Commonly in reactive programming - we don't do anything unless the data changes. Hence, we are reacting to changes - reactive programming!

RxJs in Action

```
observeTweets()
    .filter(s => s.message.contains("Greetings!"))
    .flatMap(s => getUserAvatar(s.user))
    .map(a => $(`<img />`).attr("src", a.url))
    .subscribe($avatar => { $avatars.append($avatar) };
```

Reactive extensions allows us to create a push model, and gives us tools for transforming the data and making it a lot easier to utilize that data.

So in reactive programming - we think of what data we want, what transformations we want to apply, and then what we want at the end of all those transformations so we can utilize that data for the user.

1. Getting an observable that will produce new values every time a new tweet comes in.
2. Filter out the items that do not contain the words "Greetings" in them.
3. Flatmap takes something that returns a promise or a observable, and merges it right in to the stream. Now we're dealing with whatever getUserAvatar returns. In this case, it could be a promise.
4. Map our avatar to a new element with the src as the avatar's url.
5. Then we subscribe to the transformation - the entire pipline - and what we get back is an $avatar object.