6)What are subjects/Observables vs Subjects

Slide 13

Subjects are like observable that don’t start automatically. When we subscribe to observable it runs same code again and again(each time we subscribe to observable it runs its code) but subjects do not.

Since subjects don’t starts as soon as you subscribe to them, you can have multiple subscribers which get same data. Also subjects can have various sources of data, since various things can call next method on subject.

Slide 14

Observable starts to execute as sons as someone subscribes to it. If we are making http request in angular, each time we subscribe to that observable, a new http request will be made.

Lets say we have subject 1, it has one subscriber. So when it emits a value 1, subscriber will execute its code. But lets say after some time a, new subscriber subscribe to function. now first value was emitted subscriber had not subscriber to our subject , so now value1 is gone. new subscriber is not aware of it. For every next value new both subscriber will execute their code. they will get same value from subject

But in case of observables-

var observer1 ={

    next: function(value){

    console.log(value);

    },

    error: function(error){

    console.log(error);

    },

    complete: function(){

    console.log("complete");

    }

    };

var observer2 ={

        next: function(value){

        console.log(value);

        },

        error: function(error){

        console.log(error);

        },

        complete: function(){

        console.log("complete");

        }

        };

var observable = Rx.Observable.interval(1000);

setTimeout(()=> {

    observable.subscribe(observer2);

}, 5000);

observable.subscribe(observer1);

whenever ever a new subscriber subscribes to observable, both of them will get different values. They wnt be in sync. Here wg=hen second subscriber subscribes to observable, observable starts sending it values from 0. While it continues to give values to to its previous subscriber. So both subscribers are getting different values.

But in case of subjet, old value are ignored. All subscribers at a time will get same value that is emitted by subject.

8)Behaviour Subjects/Behaviour Subject Demo

See notes of last lecture max’s behaviour subject.

9)Replay subjects/ Replay subjects Demo

Slide 18

Here we will learn about replaySubject another useful variant of subject. There is also another variant called async subject , but that does not have any use cases that are relevant. So they work like behaviour subjects, except with behaviour subject you are guaranteed to get one value ReplaySubject may give you anywhere between 0 to X value. Where X is however many value you tell the subject to remember. So its useful if new subscribers need to catch up with a stream of incoming data without necessarily needing all of it. So a good example is message board, you might be interested in last 10 values that were pushed to message board when you add yourself a new message board widget to client’s UI. The one;s before it does’nt matter because message board cutsoff after 10 messages and you you dnt want to see it empty at first and then see one , then 2 and till it finally gets 10 messages.

Code-

var rSubject = new Rx.ReplaySubject(3);

var observer = name => console.log(name + ' is great');

rSubject.subscribe(observer);

rSubject.next('Sumeet');

rSubject.next('Archana');

rSubject.next('Nitesh');

rSubject.next('Sukhmeet');

var observer2 = name => console.log(name + ' is very bad');

rSubject.subscribe(observer2);

rSubject.next('Ankur');

output –

Sumeet is great

app.js:2 Archana is great

app.js:2 Nitesh is great

app.js:2 Sukhmeet is great

app.js:8 Archana is very bad

app.js:8 Nitesh is very bad

app.js:8 Sukhmeet is very bad