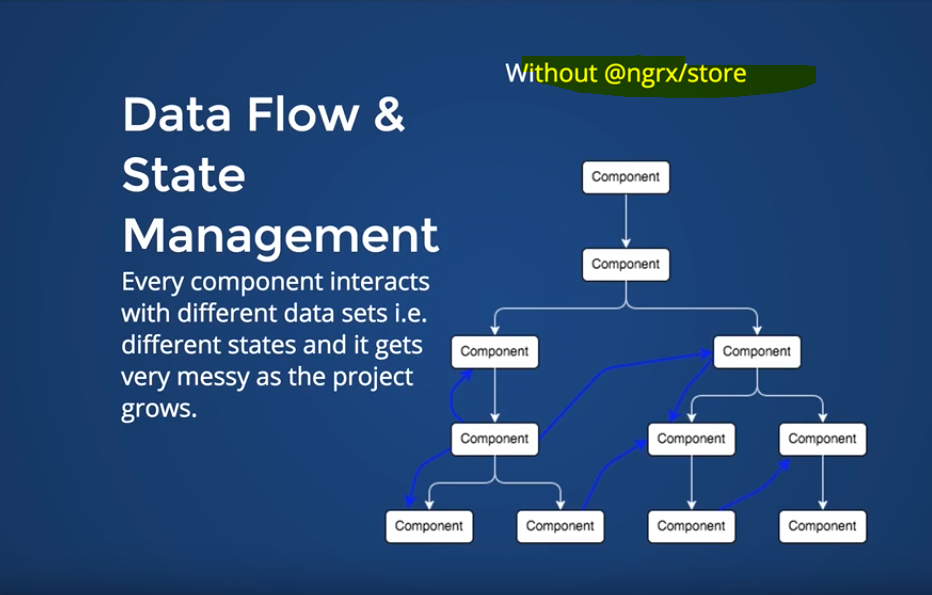
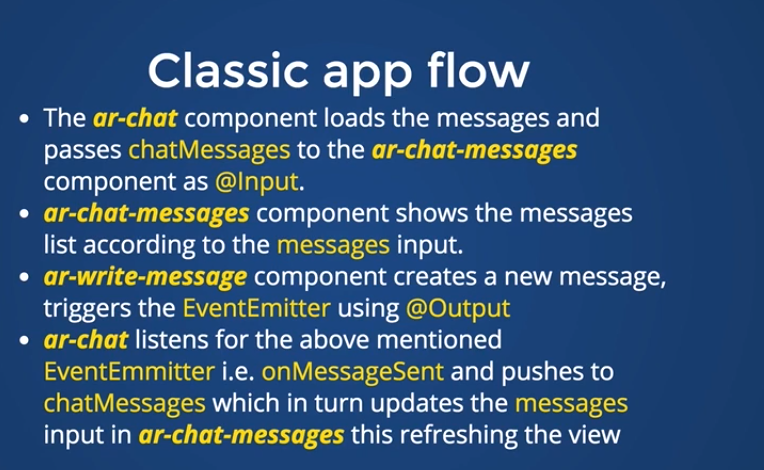
NgRX - Reactive Extensions for Angular.It is collection of reactive extensions like Ngrx/store,apex, devtools,router,db

@ngrx/Store, for ex if there are 2 sibling components and they have to communicate data with each other , we use observables, service or emit to parent and forward to child component



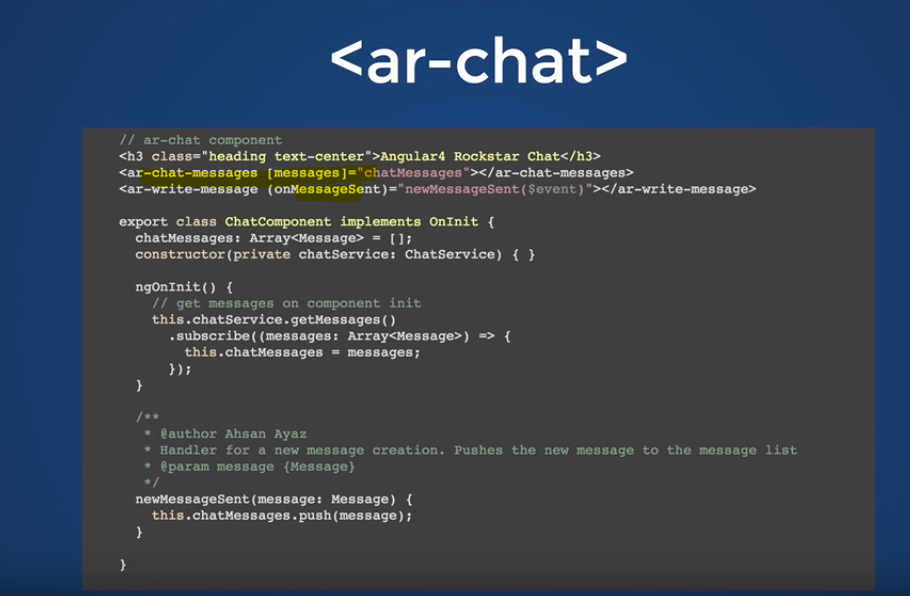
Chat app - 1. Write message component , heading comes under child component



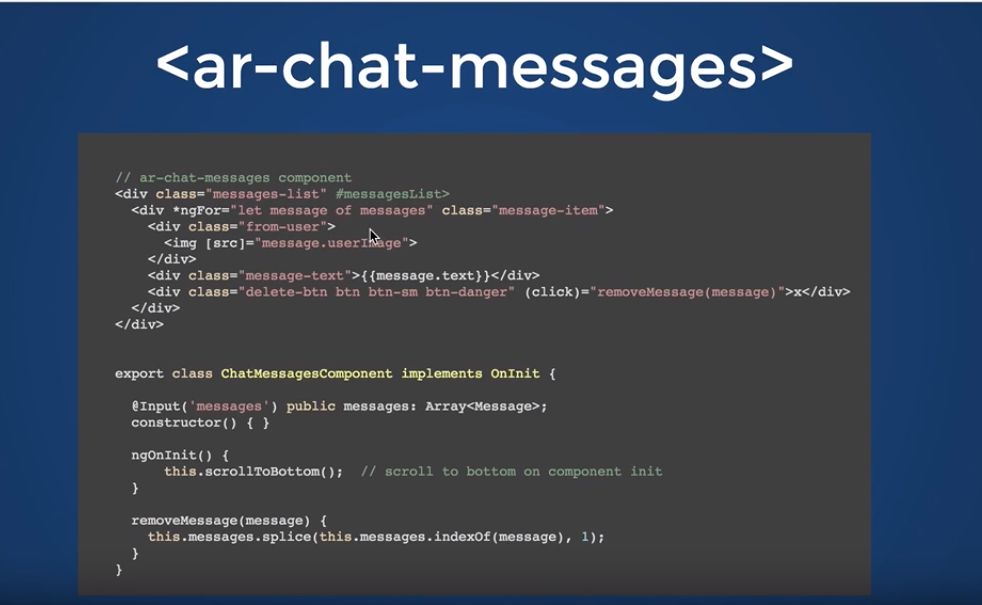


We have 2 components in (ar-chat) , chat messages and write messages

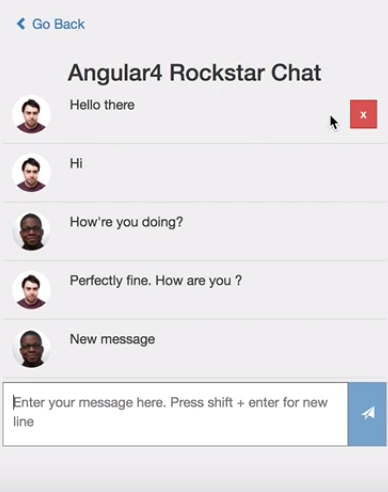
We are passing messages array with model chat messages inside , so from parent component (ar chat) we pass data into ar- chat messages and we listen for event (onMessagesent) for **write** component, so when we enter a new message it emits an event to the parent which updates the array chat messages, so chat component renders it



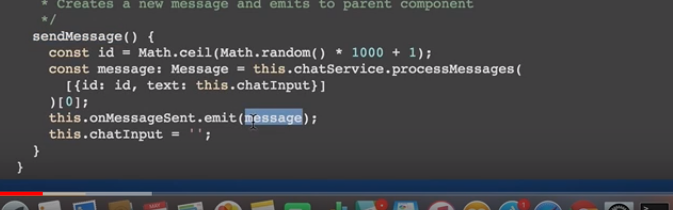
Chat messages loops through message component, remove message method removes the message when we hover through it



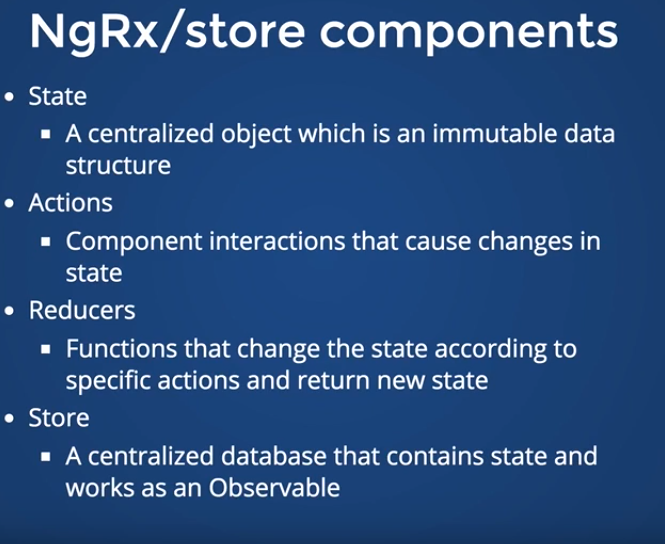
Remove message method



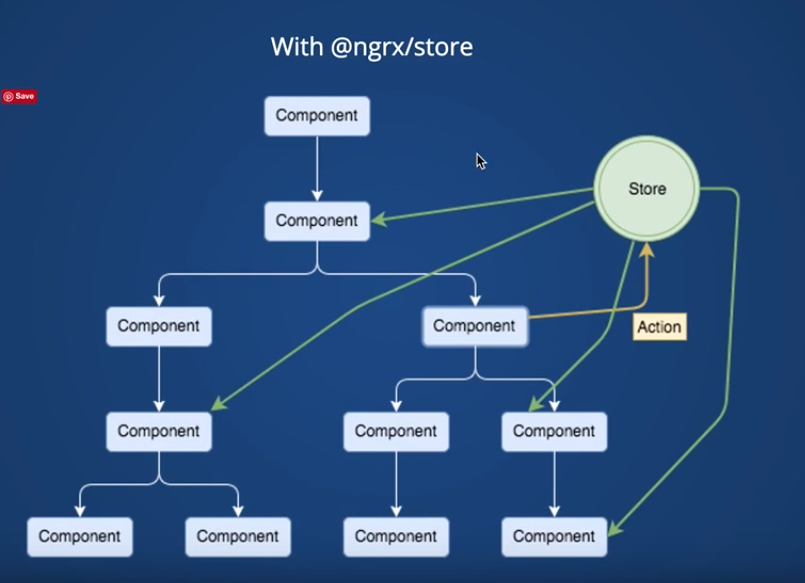
**SEND MESSAGE METHOD** - it creates a new message and then emits it



The complex app flow simplified BY NGRX/STORE



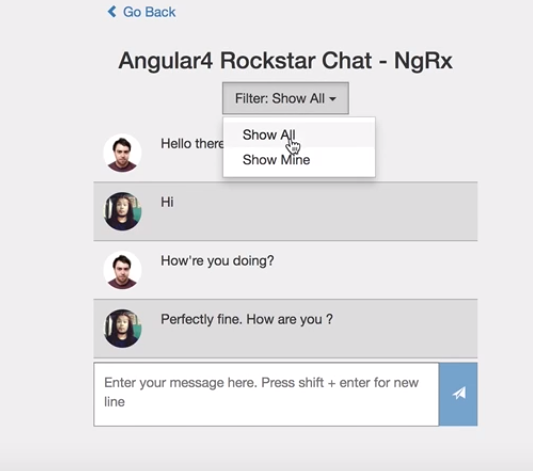
ngrx/store - when a component performs an action, the store updates every other component which relates to it



**Working with store**



We have a filter which uses ngrx/reducer which shows my or all messages

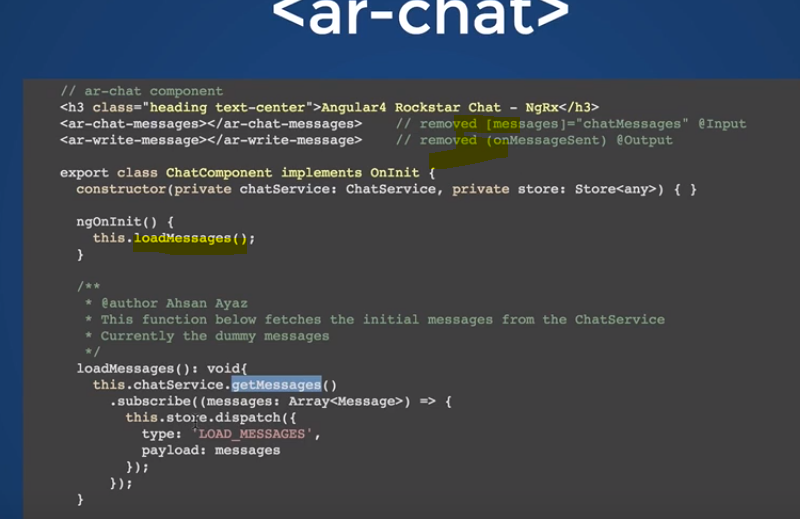


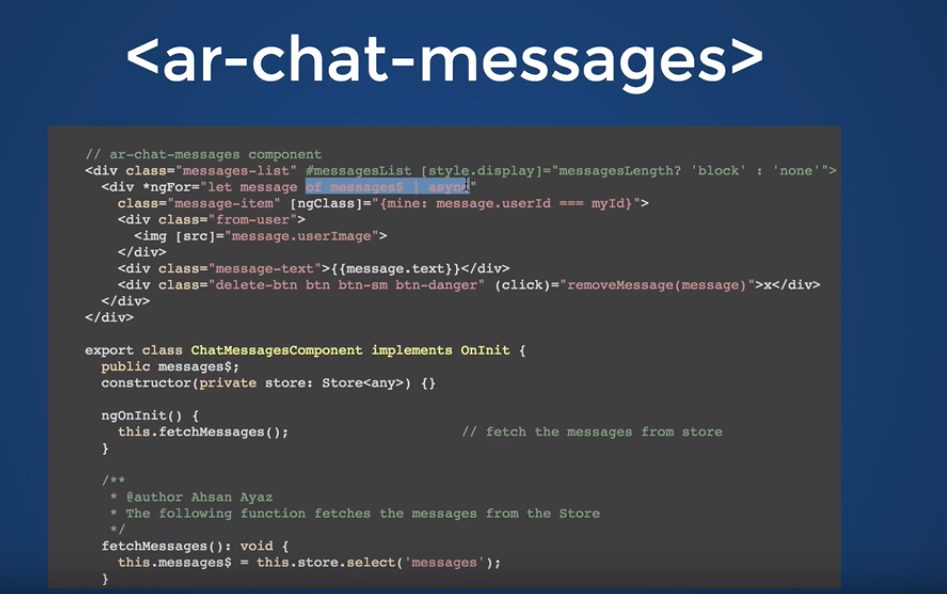
Coding with ngrx/store

We dont pass any model inside child component and we not listening to any event from child component as well

When component is initiated using ngOninit we trigger loadmessages() method which gets message from service

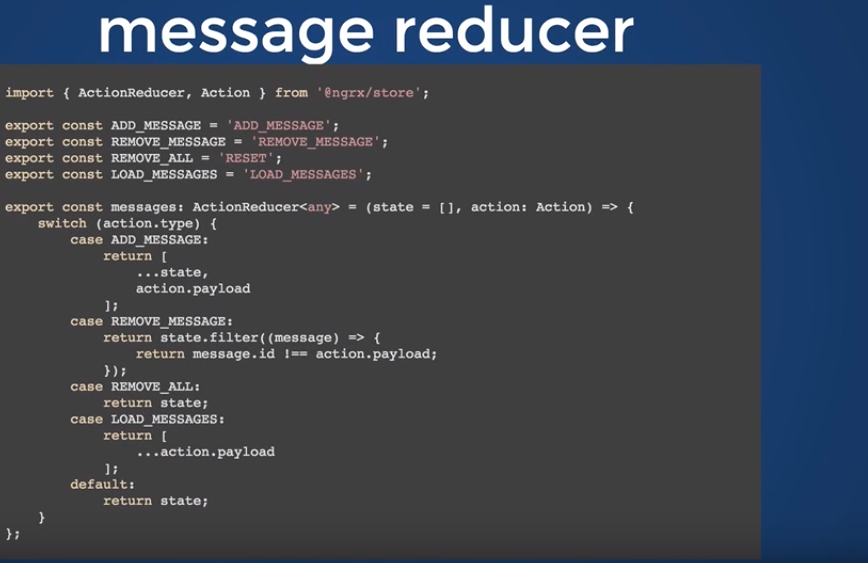
We are looping over messages model via direct observable |async , if anything changes in store we rerender the view

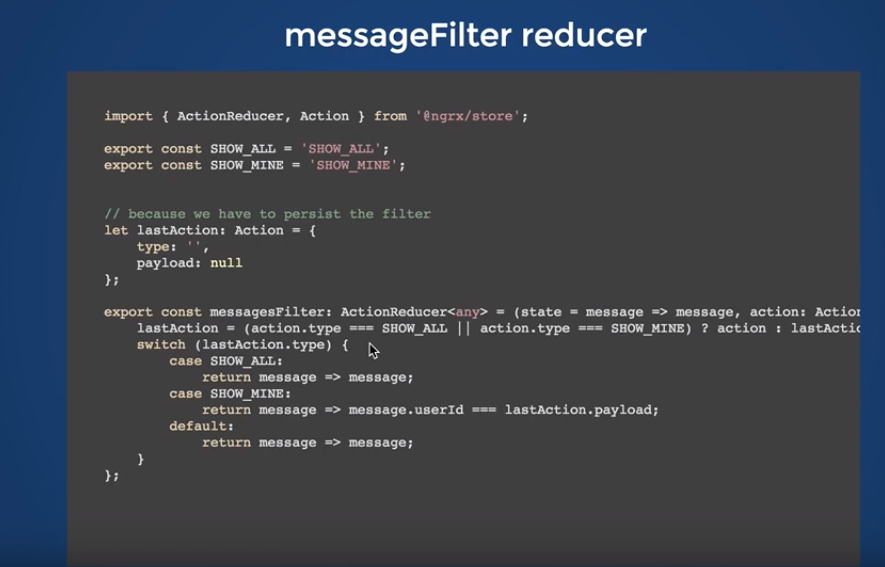




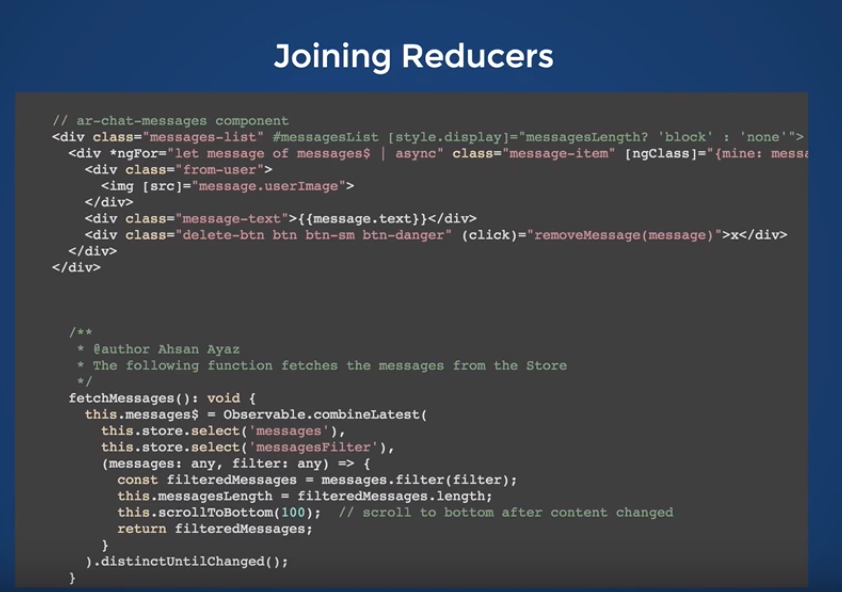


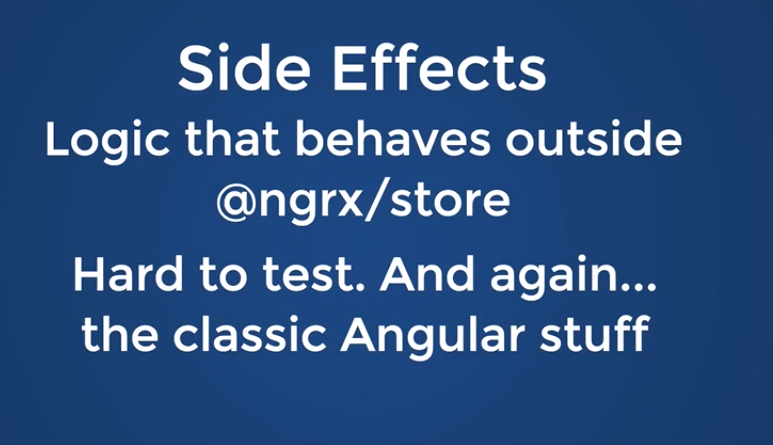
Reducer returns a new state,we dont push in previous state beacuse state is immutable

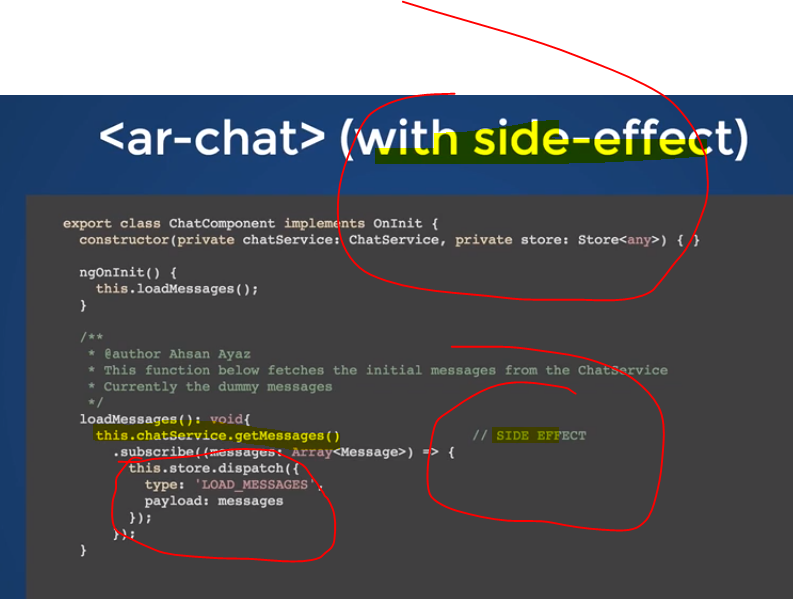




Joining reducers - one is getting messages, other is filtering

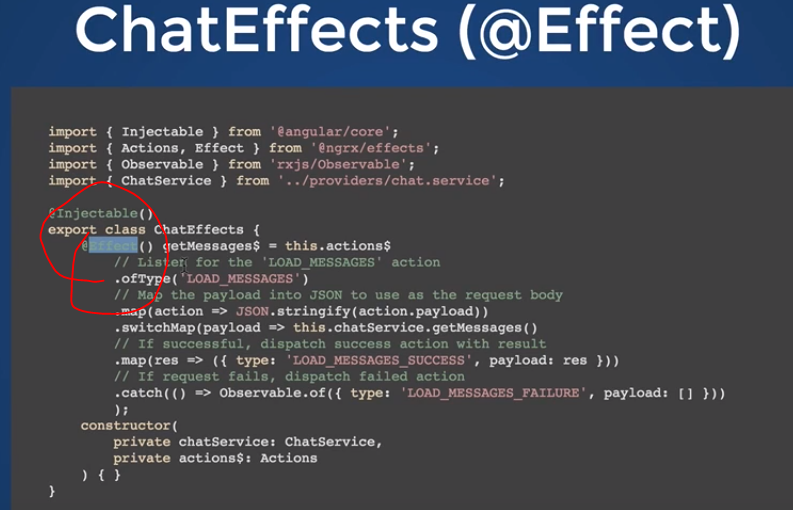


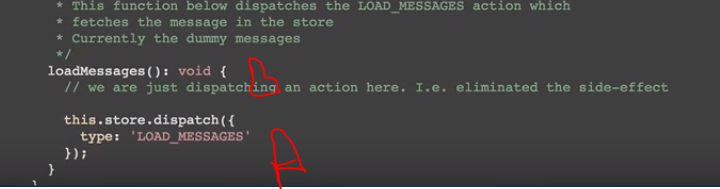




We have @effect by which we can tell store that we will dispatch an action, in which we have load message action in which

We have get messages function, if success we emit load\_message success action, else load action failure





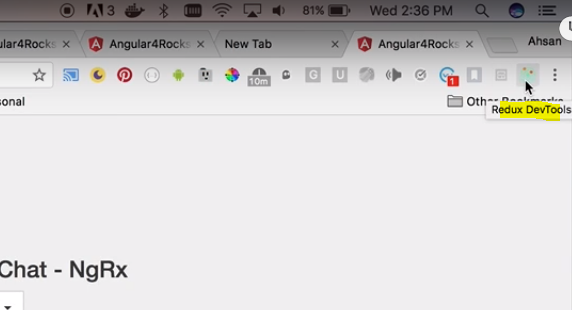




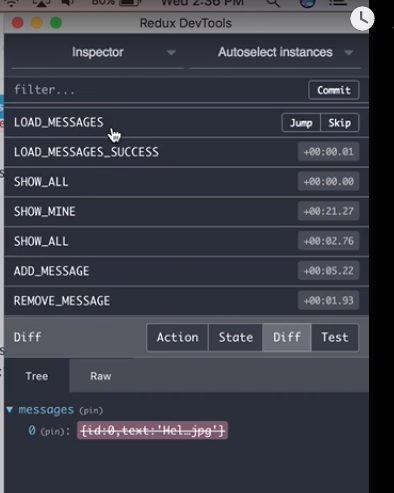
Ngrx-devtools



It turns green when we on redux/ngrx page



It contains all dispatched actions, which we can jump or skip



It has time travel debugger, if we find a bug and want to report, we press play button, and application will start playing

And whatever actions u dispatched, it will be recorded