<https://redux-observable.js.org/docs/basics/Epics.html>

# Epic

Epic is functions that takes stream (Observable) of action and returns stream of action.

Actions are run through normal reducers before it goes through epics. Hence, it is not possible to swallow the original action. However, we can dispatch a new action in epic.

If we pass through an action then it will create an infinite loop.

All action that are dispatched on redux store are next() of epic’s first input variable which is an observable.

All next() of epic output observable are dispatched to redux store.

Steps:

1. Create an epic
2. Combine epics to create a root epic(combineEpics())
3. Create an epic middleware ( createEpicMiddleware())
4. apply epic middleware to store (pas s to createStore())
5. run middleware for rootEpic

const pingPongEpic = (action$, state$) => {

    console.log('Logging state$ in ping pong epic');

    const stateSubscription = state$.subscribe(state => console.log('next state in ping pong epic:', state));

    console.log('Logging action$ in ping pong epi');

    action$.subscribe(action=> console.log('action next in ping pong epic', action));

    return action$.pipe(map(action=> {

        console.log('action in ping pong:', action);

        return action;

        }),

        ofType(PING.type),

        mapTo(PONG)

    )

}

**Log for above epic:**

Logging state$ in ping pong epic

pingPongEpic.js:11 next state in ping pong epic: {userReducer: {…}, pingPong: {…}}

pingPongEpic.js:12 Logging action$ in ping pong epi

userComponent.js:47 mapStateToProps: {userReducer: {…}, pingPong: {…}}

**After a different action(LOAD\_USER\_ACTION) is dispatched**

userReducer.js:9 in reducer for LOAD\_USER\_ACTION

userComponent.js:47 mapStateToProps: {userReducer: {…}, pingPong: {…}}

pingPongEpic.js:11 next state in ping pong epic: {userReducer: {…}, pingPong: {…}}userReducer: {user: Array(1)}user: ["asd"]\_\_proto\_\_: ObjectpingPong: {isPinging: false}\_\_proto\_\_: Object

pingPongEpic.js:13 action next in ping pong epic {type: "LOAD\_USER\_ACTION"}

pingPongEpic.js:15 action in ping pong: {type: "LOAD\_USER\_ACTION"}

**Second parameter is stream of state.**

Every time an action is dispatched on redux store, an action is returned (similar subscriber.next(action) in action$ observable) from action$.

If the store if updated by any reducer state$ returns a new state.

For every new item in action$, operators are executed.

**Note**: Here epic function is not executed again and again. It just the operators are executed for every new item in action$(rxjs operator functionality).