**STATE MANAGMENT**

Let’s talk about state management. Basically, every data which affects how UI looks like is your application state. Like a collection of records from API which is used as a data source for the main grid and to form different filtering pane for example. The range of different independent UI elements can rely on the state of the app. And at the same time, the state can be changed by different events: new data asynchronously arrived, some UI elements or user input caused new data entered or existing data changed.

The most important task for app architect in this field is to make sure application state is consistent with UI elements. No matter what platform we are talking about. If new data arrived — all dependent UI components should reflect this change. Probably for the simple application you are able to trigger some change detection or component render manually by some change. But the bigger application is, the more complex UI relationships are, the more complicated this task becomes. As a result, data flow might become unclear and messy.

**FLUX/REDUX PRINCIPLES**

Redux is one of the most popular and widely used frameworks for state management. It uses so-called flux principles, described below:

* Unidirectional data flow
* The state is a single source of truth
* The State is immutable
* Operates with items like actions, stores, views, reducers, subscriptions

Basically, with Redux developer defines what is state by creating a store, writes functions to change state by type of action, and manage who is listening for state changes. The most important idea here is that the state is immutable. We cannot change the state directly, we can only create a new one using a defined function. This means that the state is always consistent. There is no chance the state will be incorrect unless reducer function contains bugs.

Reducer is a defined function which contains logic to create a new state based on action.

You can manually subscribe or unsubscribe for state events and add own handling logic there. But for UI updates we usually use React-Redux library or similar connector, which do the same automatically.

**REDUX**

It mainly focuses on three components; store, reducer, and Provider.

**store**: It is the structure where the definitions and initial values ​​of the states needed for the relevant context are determined. It is a JS object

**reducer**: It is the function responsible for updating the specified global states.

It is a function that returns the current state value and the parameters sent while triggering as arguments.

**Provider**: Allows the created structure to be defined in the form of a custom React component. It is a React component.