Redux: Actions, Reducers, Store

What are actions?

Actions are plain javascript objects with a **type** field. In other words, you can think of an event that describes what happened in the app. Actions are the only source of information for the store. It carries a payload of information from the app to the store.
For eg. If a user puts a request to log in it is the action.
If he successfully logs in then that is also an action. And if puts a request to create a post then that is also an action.

Types should be defined as string constants in your application as given below –

const REQUEST = 'REQUEST';

What are Action Creators?

Action creators are the functions that contain the creation of action object. These functions return the plain JS object which is an action. It promotes **clean code** writing and helps achieve **reusability.**

What are pure functions?

A pure function is a process of inputting arguments and producing the same output called the **return value**. A function is said to be **pure** if it follows the following rules -

- A function gives the **same result** for the same arguments.
- There is no downside of its evaluation, i.e. it does not change the input data.

- No **mutation** of local and global variables.
- It does not depend on an **external state** such as a global variable.

What are Reducers?

Reducers are pure functions in Redux. Pure functions can be predicted easily. Reducers are the only way to change states in **Redux**. This is the only place where you can write calculations and logic. The reducer function accepts the previous state of the application and the action calculates the next state and returns the new object.

Reducers should never do the following -

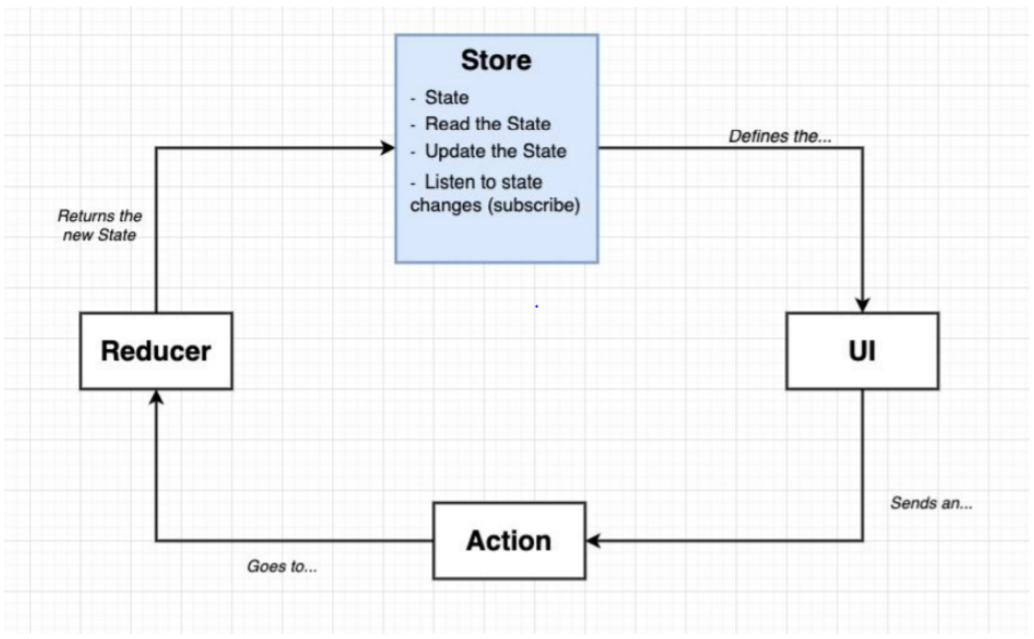
- Mutation of functional arguments
- API call and routing logic
- Calling non-pure function e.g. Math.random()

The following is the syntax of a reducer −

(state,action) => newState

What is Store?

The **store** is the immutable object tree in **Redux**. The store is the state container that holds the state of the application. Your app only has one store of Redux. Whenever you create a store on Redux, you must specify the reducer.



createStore(): createStore function is the inbuilt function in Redux that helps us to create a store for our application.

```
import { createStore } from 'redux';
import reducer from './reducers/reducer'
const store = createStore(reducer);
```

createStore function has three arguments:

- **Reducer:** This returns the next state of the application.
- **preloadedState:** This is an optional argument and contains the initial state of the app.
- **Enhancer:** It will help to enhance the app with third-party capabilities. It is also an optional argument.

Some important functions associated with the store:

- **Dispatch:** This function allows to dispatch an action from the store to change state in your application.
  Syntax: *store.dispatch({type:'ITEM'})*

- **getState:** This function allows you to fetch the current state of your application.
  Syntax: *store.getState()*

- **Subscribe**: This function allows you to create a **callback** function that Redux will call when your current action is dispatched. As soon as the new state gets updated it re-render the view automatically.
  Syntax: *store.subscribe(()=>{ console.log(store.getState());})*

  For example: if we are loading data from the database and it takes time then the screen shows LOADING... This is called subscribing.

Note: We can also unsubscribe something using store.unsubscribe().

Some Important Functions:

- **indexOf:** This function iterates over a string or an array and find out the specified value.
- **Filter function:** The filter function basically filters values on the basis of condition and stores that value in the array.
- **Switch:** The switch statement executes the expression according to the case clause and also executes the expressions present under that case.

  Syntax:*switch (expression) {*

  ```
  case value1:
    [break;]
  case value2:
    [break;]
  ...
  case valueN:
  ```

```
    [break;]
  [default:
    [break;]]}
```

Summarising It

Let’s summarise what we have learnt in this module:

- Learned about actions and pure functions in detail.
- Learned about Redux store and its important functions.
- Learned about reducers in detail.

Some References:

● Pure functions
https://www.freecodecamp.org/news/what-is-a-pure-function-in-javascript-acb887375dfe/
● Actions
https://redux.js.org/basics/actions
● Reducers
https://redux.js.org/basics/reducers
● Store
https://redux.js.org/api/store