Redux

Predictable State Container

What is Redux

Redux is a predictable state container for JavaScript apps.

Redux - Principles

- Single source of truth
- State is read-only
- Changes are made with pure functions

Redux - Core Concepts

- The state of your whole application is stored in an object tree within a single **store**
- The only way to change the state is to emit an **action**, an object describing what happened
- To specify how the state tree is transformed by actions, you write pure **reducers**.

Redux - Store

- The state of your whole application is stored in an object tree within a single **store**
- The only way to change the state is to emit an **action**, an object describing what happened
- To specify how the state tree is transformed by actions, you write pure **reducers**.

Redux Action

- The *only* way to change the *state* is to emit an ***action*****,** an object describing what happened

```
{
    type : ‘MAKE_CHOCOLATE’,
    ingredients : ['Chocolate Liquor','Cocoa Butter','Sugar','Milk']
}
```

Redux - Reducer

- To specify how the *state* tree is transformed by *actions*, you write ***pure reducers***.

```
function reducer(prevState = {counter: 0}, action) {
    let newState = prevState;
    if (action.type === 'INC') {
        newState = {counter: prevState.counter + 1};
    }
    if (action.type === 'DEC') {
        newState = {counter: prevState.counter - 1};
    }
    return newState;
}
export default reducer;
```

Redux - Setup

```
npm install redux --save
```

Redux - In Action

- Plan actions
- Plan reducers
- Create store

Redux - In Action - Plan Actions

- User clicks INC
- User clicks DEC

```
{type: “INC", by:5}
```

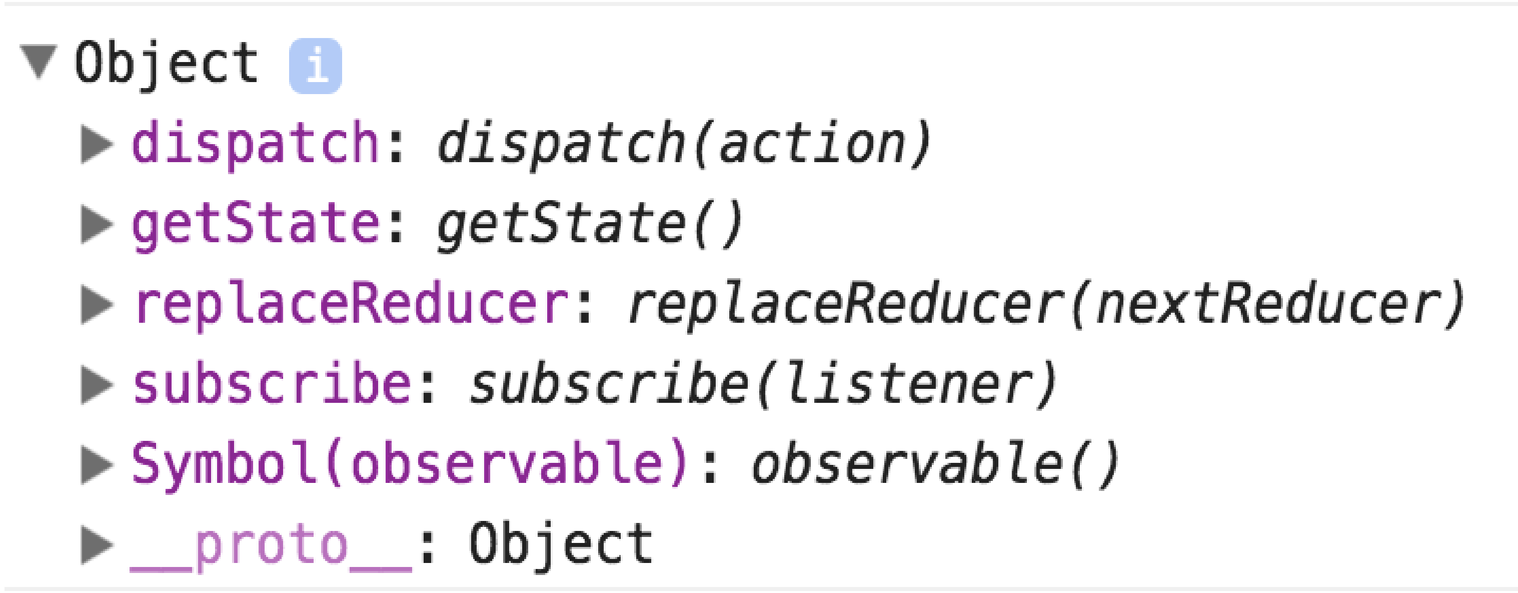
An action is an JSON object describing intent of action with type key and supporting payload if any

Redux - In Action - Plan Reducers

```
import * as UserActions from '../actions/UserActions';
const initialState = {users: []};
export default (prevState = initialState, action) => {
            switch (action.type) {
            case UserActions.USER_SIGNUP:

            let users = prevState.users;

            users.push(action.user);

            return Object.assign({}, prevState, {users}, {userSignedUp: true});   ⟵
            case UserActions.USER_LOGIN:

             let loggedInUser = prevState.users.filter(user => user.email === action.user.email && user.password === action.user.password);

            return Object.assign({}, prevState, {isAuthenicated: loggedInUser.length > 0 ? true : false});   ⟵
            default:

            return prevState;

      }

}
```

Redux - In Action - Plan Store

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



```
▼ Object
  ▶ dispatch: dispatch(action)
  ▶ getState: getState()
  ▶ replaceReducer: replaceReducer(nextReducer)
  ▶ subscribe: subscribe(listener)
  ▶ Symbol(observable): observable()
  ▶ __proto__: Object
```