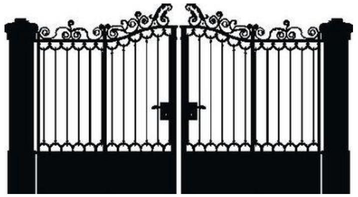
React js Lecture 4

By Mona Soliman

Agenda

- Recap last lecture points
- Interceptors
- What is Redux ?
- Redux components
- Getting started with store
- Useful extensions
- Questions!

Interceptors



Axios interceptors are the default configurations that are added automatically to every request or response that a user receives. It is useful to check response status code for every response that is being received.

```
axios.interceptors.request.use
(
    (req) => {
        // Add configurations
here
        return req;
    },
    (err) => {
        return
Promise.reject(err);
    }
);
```

```
axios.interceptors.response.use(
    (res) => {
        // Add configurations here
        if (res.status === 201) {
            console.log('Posted
Successfully');
        }
        return res;
    },
    (err) => {
        return Promise.reject(err);
    }
);
```

REDUX

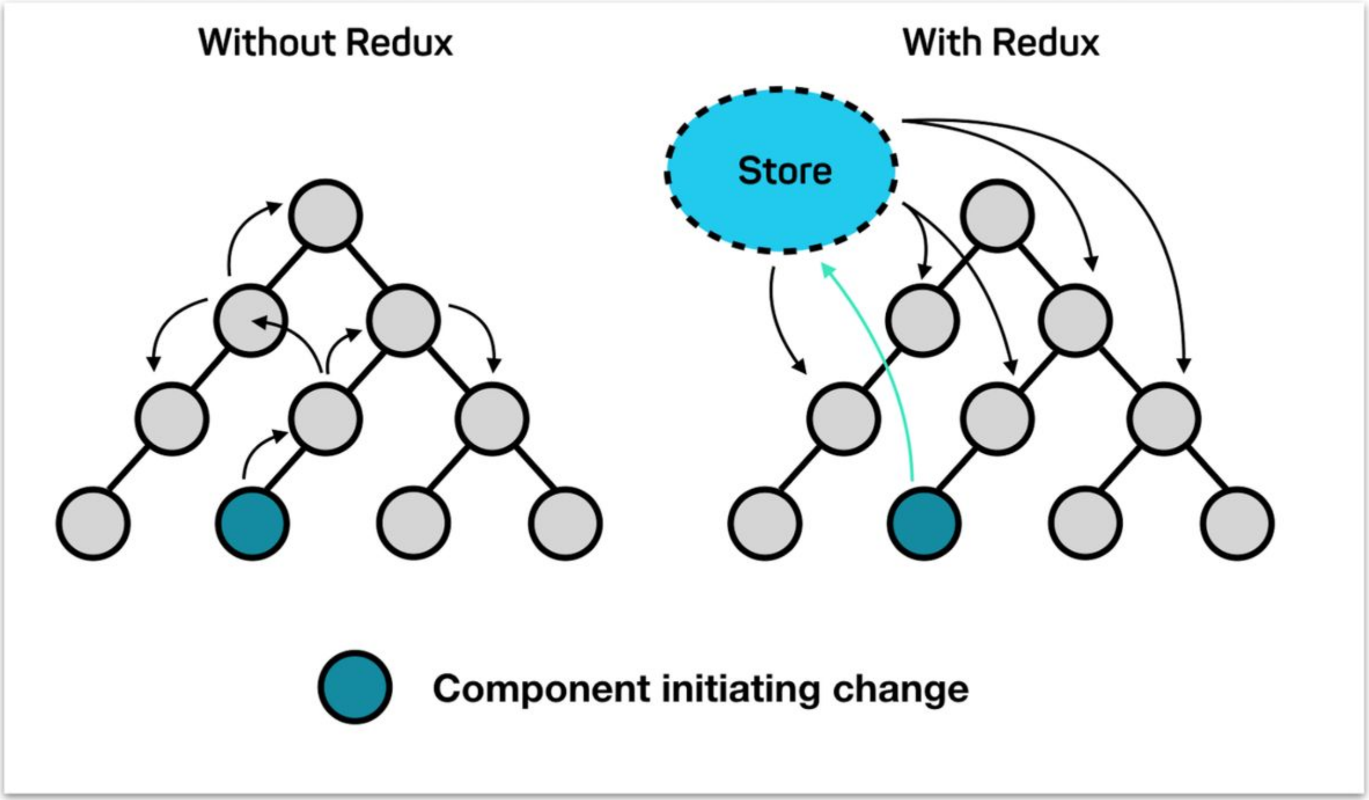
Redux is a predictable state container for JavaScript apps.You can use Redux together with React, or with any other view library.the state of your application is kept in store. And any component can access the state from the store.

To use redux in your application :

Npm install redux react-redux

https://redux.js.org/introduction/getting-started

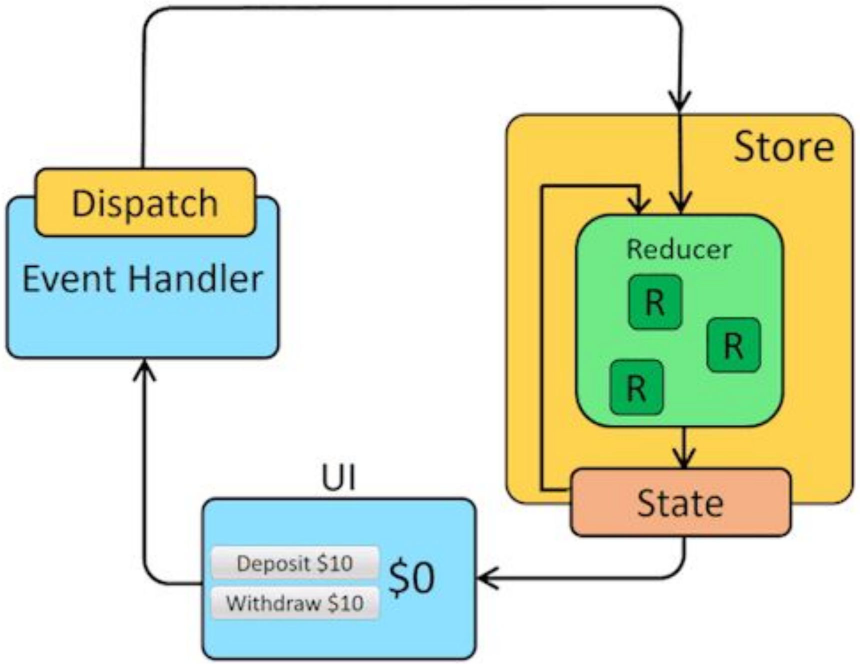
REDUX


Without Redux
With Redux
Store
Component initiating change

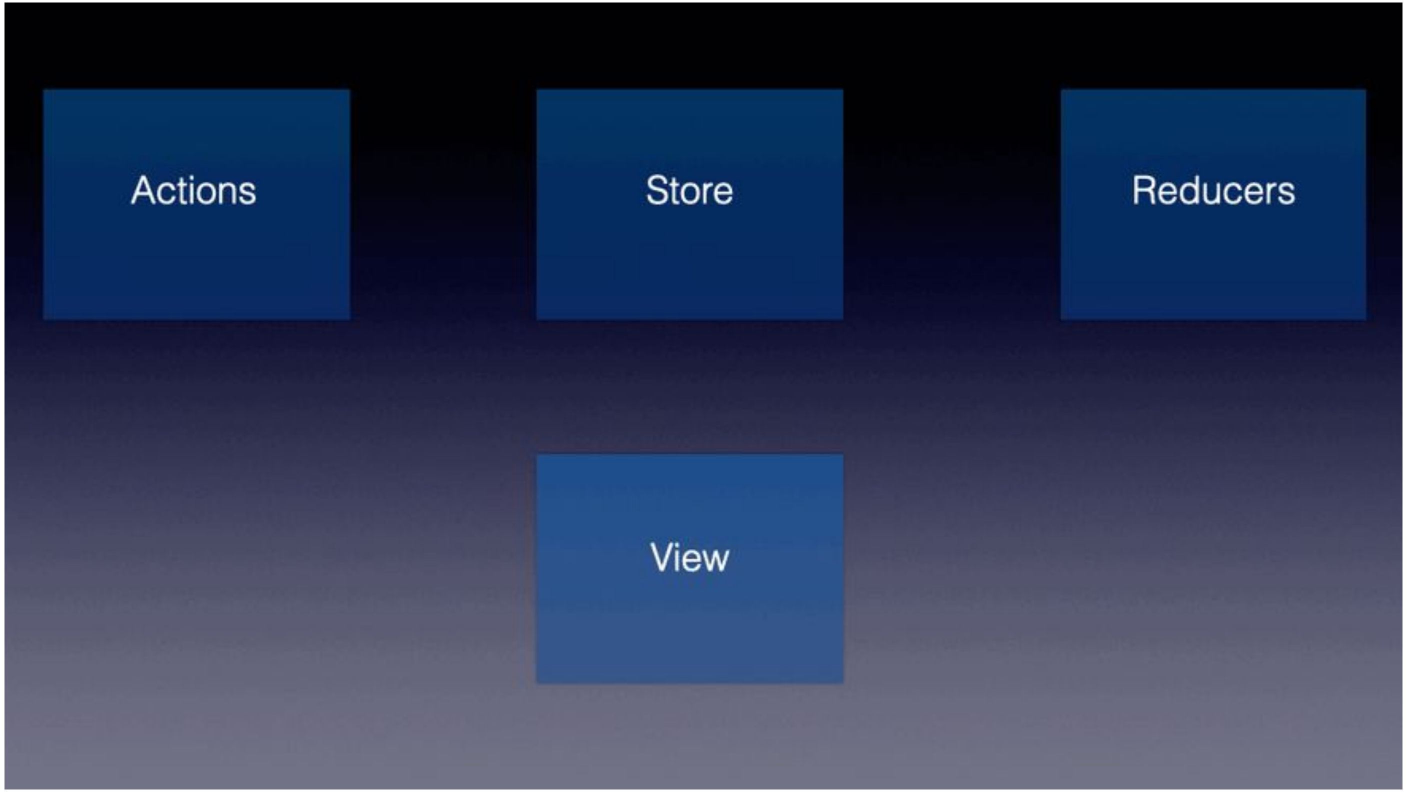
REDUX

There are three building parts :

- Actions
- Reducers
- Store



REDUX



Redux actions

these are objects that should have two properties, one describing the **type** of action, and one describing what should be changed in the app state.

```
const setUsername = (payload) => {
return {
type: "LOGIN",
payload: payload
}
}
```

Redux reducers

these are functions that implement the behavior of the actions. They change the state of the app, based on the action description and the state change description.

```
export default (state = {}, action) => {

switch (action.type) {

case type:

return {

...state,

...action.payload,

};

default:

return state;

}
```

Redux Store

The Redux store brings together the state, actions, and reducers that make up your app. The store has several responsibilities:

- Holds the current application state inside
- Allows access to the current state via store.getState();
- Allows state to be updated via store.dispatch(action);
- Registers listener callbacks via store.subscribe(listener);
- Handles unregistering of listeners via the unsubscribe function returned by store.subscribe(listener).

Redux Store

```
import { createStore} from "redux";

import reducers from "./reducers";

//redux dev tool

const composeEnhancers =

window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__();

const store = createStore(reducers , composeEnhancers );

export default store;
```

```
Or install redux-devtools-extension ,

import { composeWithDevTools } from 'redux-devtools-extension'

Then pass composeWithDevTools() instead of composeEnhancers
```

Wrap your app with redux

```
<Provider store={store}>

<App />

</Provider>
```

Read And Update Store in Functional Components😀

Read and update store(functional component)

UseSelector()

To read and get different store values

Syntax : const state = useSelector(state => state)

UseDispatch()

To Update store values and dispatch actions

Syntax : const dispatch = useDispatch();

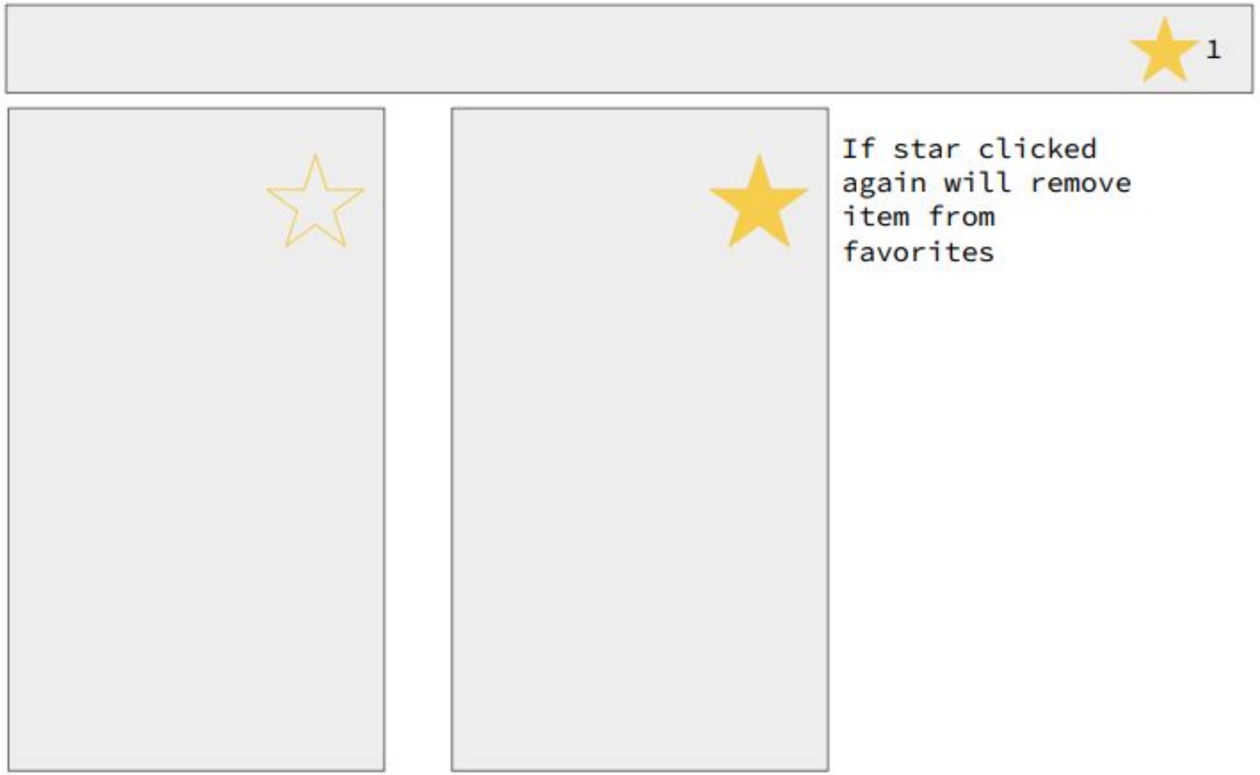
dispatch(action());

LAB >> 🌸

Create redux cycle to add movies to favorites:

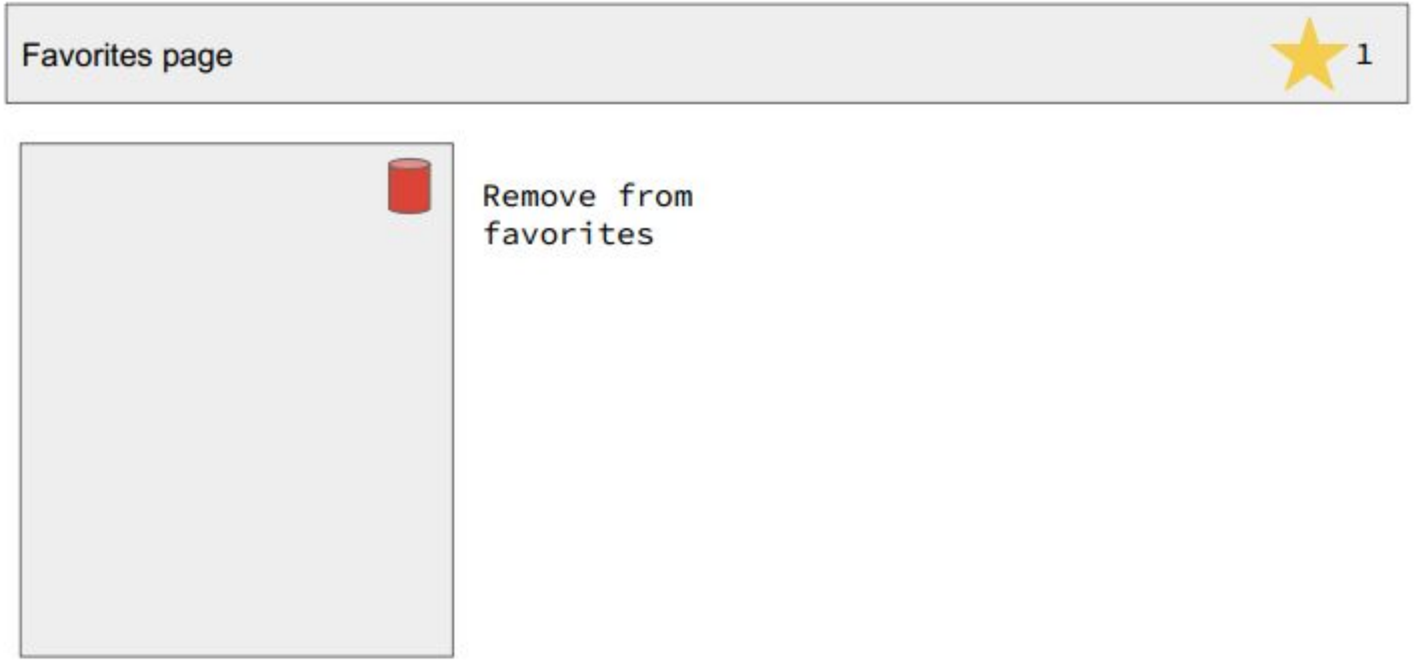
- if movie added to favorites star or heart icon should be styled with filled color.
- If movie not in the favorites icon should be bordered.
- User can go to favorites page
- Favorites count should appear in navbar
- User can remove movies from favorites page

Lab>>



If star clicked again will remove item from favorites

Lab>>



Thank you