React.js

Lecture 4

Agenda

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

Redux

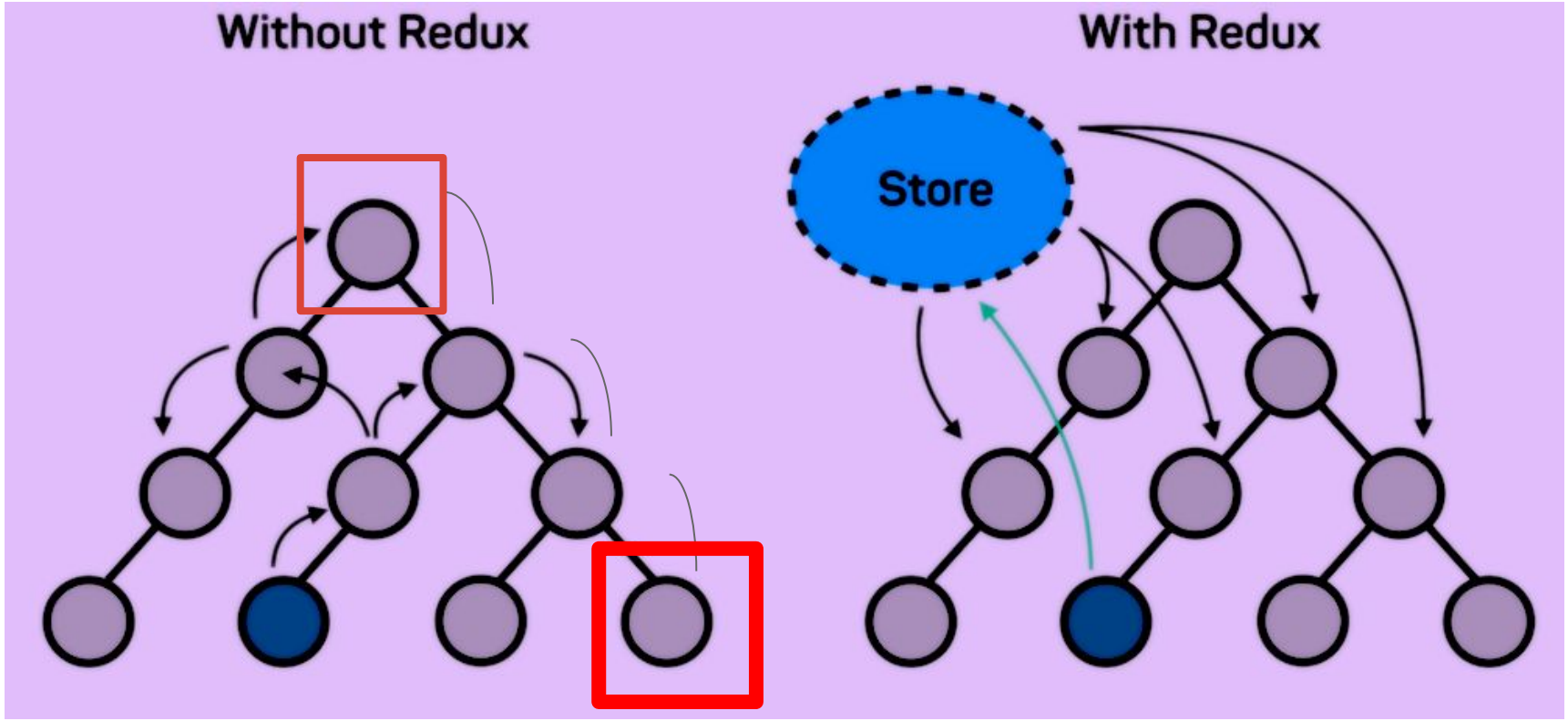
Redux is a predictable state container and the state of your application is kept in a store, and each component can access any state that it needs from this store.

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

To use Redux in your react app you need to install it first :

npm install redux react-redux

Redux

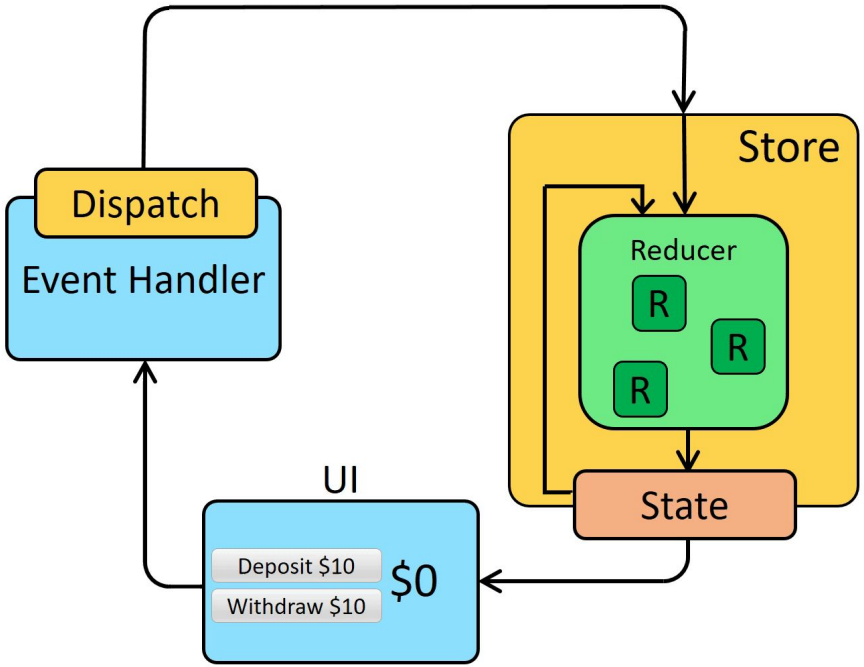


How Redux works

There is a central store that holds the entire state of the application. Each component can access the stored state without having to send down props from one component to another.

There are three building parts:

- actions
- store
- reducers



How Redux works : Actions

Actions are plain JavaScript objects, and they must have a **type** property to indicate the type of action to be carried out.

Example :

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

How Redux works : Reducers

Reducers are pure functions that take the current state of an application, perform an action, and return a new state. These states are stored as objects, and they specify how the state of an application changes in response to an action sent to the store.

How Redux works : Reducers

```
import * as types from "./types";

export default (state = {}, action) => {
  switch (action.type) {
    case type:
      return {
        ...state,
        ...action.payload,
      };
    default:
      return state;
  }
};
```

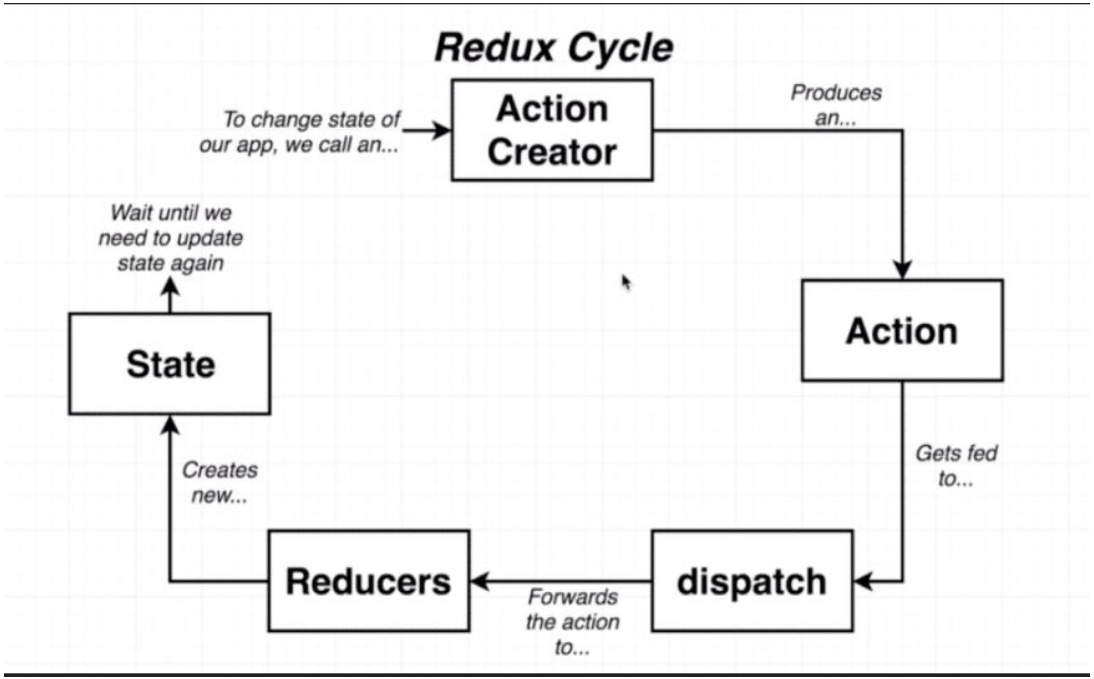
How Redux works : Reducers

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

export default combineReducers({
  lang,
});
```

How Redux works : Store

The store holds the application state. There is only one store in any Redux application.



How Redux works : 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

How Redux works : Wrap application with Redux

```
 <Provider store={store}>
  <App />
</Provider>
```

Let's try redux

With react app

Read And Update Store in Functional Components

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());
```

Redux with class component

Connect

The connect() function connects a React component to a Redux store.

mapStateToProps

the new wrapper component will subscribe to Redux store updates. This means that any time the store is updated, mapStateToProps will be called

mapDispatchToProps

To specify and dispatch actions :

```
const mapDispatchToProps = dispatch => {
  return {
    // dispatching plain actions
    actionName: () => dispatch(actionName()),
  }
}
```

Useful extensions

- React Developer Tools
- Redux DevTools

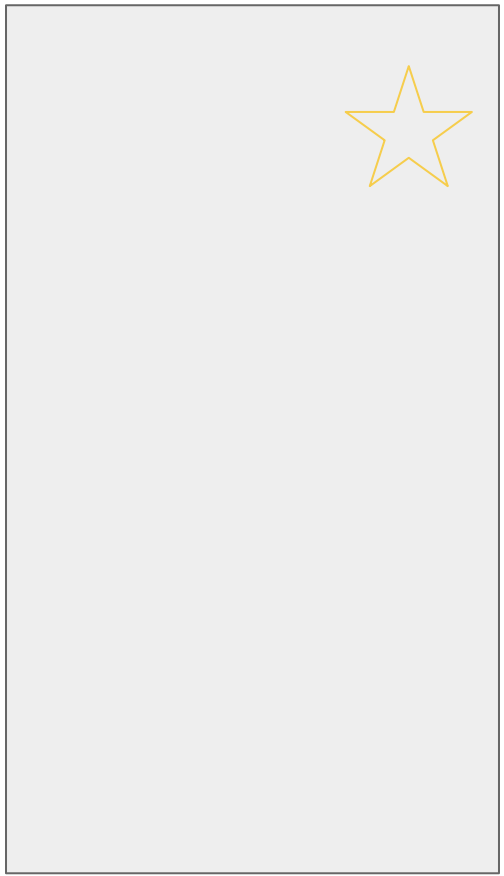
Thank you

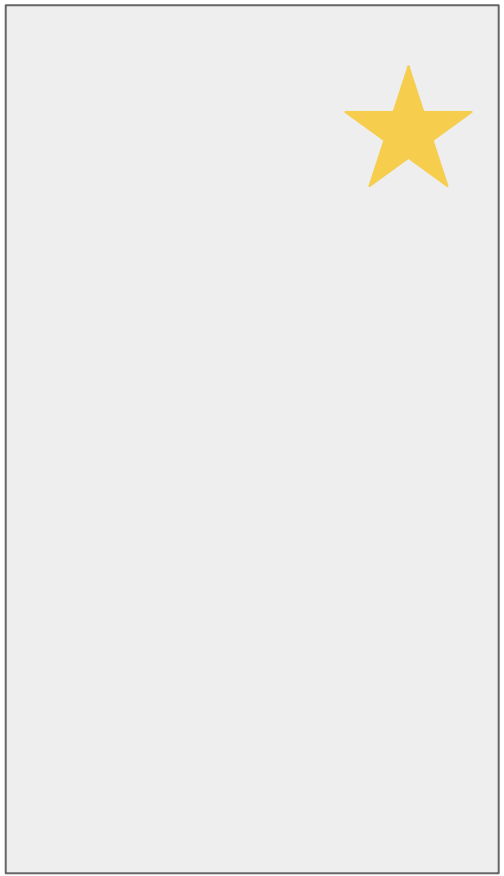
Lecture 4
Lap

Movie app :Add to favorites

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





If star clicked again will remove item from favorites

Favorites page

1

Remove from favorites