React JS

LECTURE-3

Redux

- Redux helps us with the state management.
- Creates a store that any component can used to access data.

3 Major things to keep in mind.

- Action
- Reducer
- Store
- createStore()

Action

- {type: ‘counter/increment’}
- {type: ‘counter/decrement’}

- Create a function for this, to create an ACTION.
  - Function(){
    return {
    {type: ‘counter/increment’}
    }
    }

Reducer

```
function counterReducer(state = { value: 0 }, action) {
      switch (action.type) {
            case 'counter/incremented’:
                  return { value: state.value + 1 }
            case 'counter/decremented’:
                  return { value: state.value - 1 }
            default:
                  return state
      }
}
```

STORE

- Stores are centralized and anyone can use them, as per need
- States are immutable
  - dispatch function update the states.

Installation

- npm i redux react-redux

ACTIONS

- function increment(){
  return {
  {type: ‘counter/increment’}
  }
  }
- function decrement(){
  return {
  {type: ‘counter/decrement’}
  }
  }

Reducer: Initial State

```
// Define an initial state value for the app
const initialState = {
  value: 0
}
```

We define the initial state first.

```
// Create a "reducer" function that determines what the new state
// should be when something happens in the app
function counterReducer(state = initialState, action) {
  // Reducers usually look at the type of action that happened
  // to decide how to update the state
  switch (action.type) {
    case 'counter/incremented':
      return { ...state, value: state.value + 1 }
    case 'counter/decremented':
      return { ...state, value: state.value - 1 }
    default:
      // If the reducer doesn't care about this action type,
      // return the existing state unchanged
      return state
  }
}
```

Reducer: Function

Combining Multiple Reducers

```
import { combineReducers } from 'redux'

import countReducer from ‘file_location'
import productReducer from ‘file_location'

const rootReducer = combineReducers({
  // Define a top-level state field named `counter`, handled by `countReducer`
      countReducer,
      productReducer
})

export default rootReducer
```

Creating the store!

```
import { createStore } from 'redux'
import rootReducer from './reducer'

const store = createStore(rootReducer)

export default store
```

Subscribe to the store

```
// Omit existing React imports, get the store file now!!
import store from './store'

// Every time the state changes, log it
// Note that subscribe() returns a function for unregistering the listener
const unsubscribe = store.subscribe(() =>
    console.log('State after dispatch: ', store.getState())
  )
```

Using the store: Provider

- import { Provider } from 'react-redux'

```
ReactDOM.render(
// Render a `<Provider>` around the entire `<App>`,
// and pass the Redux store to it as a prop
   <React.StrictMode>
         <Provider store={store}>
               <App />
         </Provider>
   </React.StrictMode>,
document.getElementById('root')
)
```

useSelector and useDispatch

- useSelector will help us to get the initial value of our countReducer.
- const stateValue = useSelector(state => state.countReducer);
- Now, we just need to dispatch the events that we have defined.
  - const dispatch = useDispatch()
  - onClick=()=>{dispatch(increment())}
  - onClick=()=>{dispatch(decrement())}
- That’s all about redux…