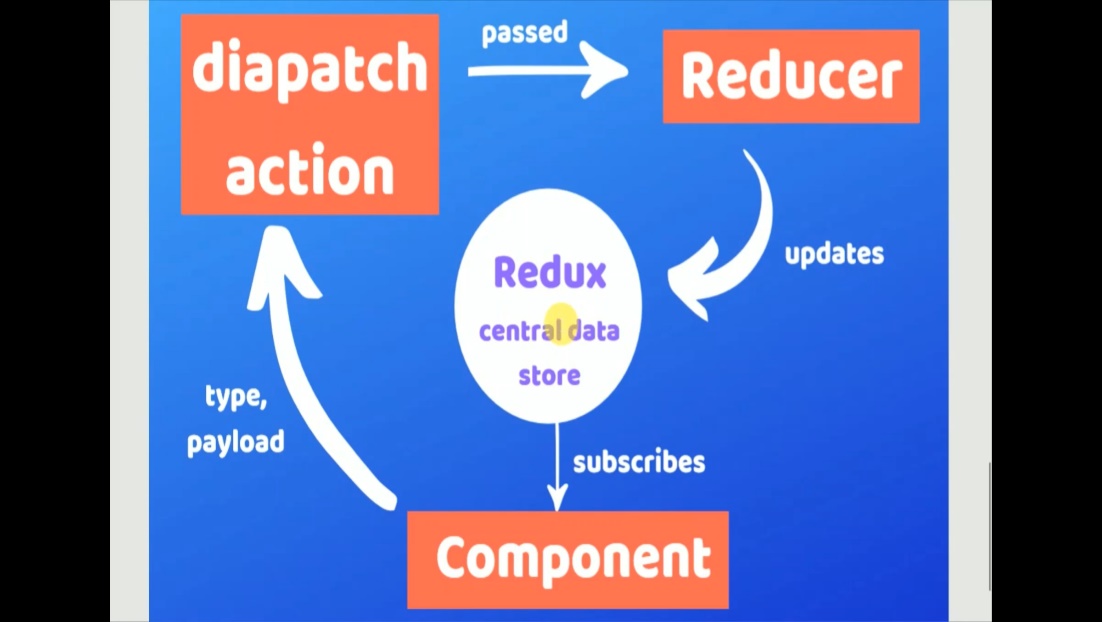
**Redux**



* **npm i react-redux redux redux-thunk**

1. Firstly, make store(Global state) by using createStore().
2. Wrap the reducer and applyMiddleware inside the store.
3. If we fetch the data through the api so use redux-thunk by using applyMiddleware.
4. Wrap the main App component inside the Provider.
5. Apply store to the Provider.

**Index.js:**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { createStore, applyMiddleware } from 'redux'

import { Provider } from 'react-redux'

import reducer from './reducers/reducer';

import thunk from 'redux-thunk'

const store = createStore(reducer, applyMiddleware(thunk))

ReactDOM.render(

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

  , document.getElementById('root')

);

reportWebVitals();

* If we want to see the redux(state,action,type..) in chrome extension so follow this Url’s step.

<https://github.com/zalmoxisus/redux-devtools-extension>

1. Create reducer:

**reducer.js:**

const iState = {

    "surname": "kachhadiya",

    "name": "pradip",

    "loder": false

}

const reducer = (state = iState, action) => {

    if (action.type === 'CHANGE\_NAME') {

        return {

            ...state,

            'name': action.payload,

            "loder": false

        }

    }

    if (action.type === 'CHANGE\_SURNAME') {

        return {

            ...state,

            'surname': action.payload

        }

    }

    return state

}

export default reducer

* Reducer contain two parameter: 1] state 2] action
* Here state is a initial(default) state.
* Reducer function is return initial or updated state.
* Action contain type and payload.
* Type means specific task to perform change the state and payload is data to set in the global state or update the initial state.
* Use switch condition(recommended)

1. Use this global state inside any component:

**App.js:**

import './App.css';

import { connect } from 'react-redux'

import { anothername } from './action';

import { useState } from 'react';

import Mix from './Mix';

function App() {

  const [loder, setLoder] = useState(false)

  return (

    <>

      <div className="App">

        <h1>Name is {loder ? "Data Fatching....." : props.name}</h1>

        <h1>Surname is {props.surname}</h1>

        <button onClick={() => { props.changename(setLoder) }}>Change Name</button>

        <button onClick={() => { props.changesurname() }}>Change Surname</button>

        {/\* <Mix /> \*/}

      </div>

    </>

  );

}

const mapStateToPropes = (state) => {

  return {

    "surname": state.surname,

    "name": state.name,

  }

}

const mapDispatchToPropes = (dispatch) => {

  return {

    changename: (setLoder) => dispatch(anothername(setLoder)),

    changesurname: () => dispatch({ type: 'CHANGE\_SURNAME', payload: 'kp' })

  }

}

export default connect(mapStateToPropes, mapDispatchToPropes)(App);

* mapStateToPropes return the global state so we can use this state in component by using props.
* mapDispatchToPropes return dispatch function.
* By using this dispatch function we can change or update the global state related to type.
* connect is H.O.C . which connect our component to the reducer..
* anothername(setLoder)) is wrapper function(action)

so we can separate them like that:

**action.js:**

// export const anothername = (name) => {

//     return {

//         type: 'CHANGE\_NAME',

//         payload: name

//     }

// }

export const anothername = (setLoder) => {

    return (dispatch) => {

        setLoder(true)

        fetch('https://jsonplaceholder.typicode.com/users')

            .then(res => res.json())

            .then(res2 => {

                dispatch({

                    type: 'CHANGE\_NAME',

                    payload: res2[2].name

                })

                setLoder(false)

            })

            .catch(err => alert(err.message))

    }

}

* Here this anothername func. take a dispatch parameter.

**Combine Many Reducer**

**Index.js:**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { createStore, applyMiddleware, combineReducers } from 'redux'

import { Provider } from 'react-redux'

import reducer from './reducers/reducer';

import thunk from 'redux-thunk'

import name\_reducer from './reducers/name\_reducer';

import wish\_reducer from './reducers/wish\_reducer';

const masterReduser = combineReducers({

  name: name\_reducer,

  wish: wish\_reducer

})

// const store = createStore(reducer, applyMiddleware(thunk))

const store = createStore(masterReduser, { name: "Pradip" }, applyMiddleware(thunk))

ReactDOM.render(

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

  , document.getElementById('root')

);

reportWebVitals();

* combineReducer is combine many reducer and provide global state…

*create diff. reducer:*

**name\_reducer.js:**

const name\_reducer = (state = [], action) => {

    if (action.type === 'CHANGE\_NAME') {

        return action.payload

    }

    return state

}

export default name\_reducer

**wish\_reducer.js:**

const wish\_reducer = (state = ["gn", "gm", "ga"], action) => {

    return state

}

export default wish\_reducer

**App.js:**

import './App.css';

import { connect } from 'react-redux'

import Mix from './Mix';

function App() {

  return (

    <>

      <div className="App">

        <Mix />

      </div>

    </>

  );

}

export default App;

**Mix.js:**

import React from 'react'

import { connect } from 'react-redux'

const Mix = (props) => {

    return (

        <div>

            <h1>Mixed Reducer</h1>

            <p><b>Name : </b>{props.name}</p>

            <p>Wishes : {props.wish.map(item => (<h6>{item}</h6>))}</p>

            <button onClick={props.changename}>Change Name</button>

        </div>

    )

}

const mapStateToPropesmix = (state) => {

    return {

        "name": state.name,

        "wish": state.wish

    }

}

const mapDispatchToPropesminx = (dispatch) => {

    return {

        changename: () => dispatch({ 'type': 'CHANGE\_NAME', 'payload': 'prk' }),

    }

}

export default connect(mapStateToPropesmix, mapDispatchToPropesminx)(Mix);