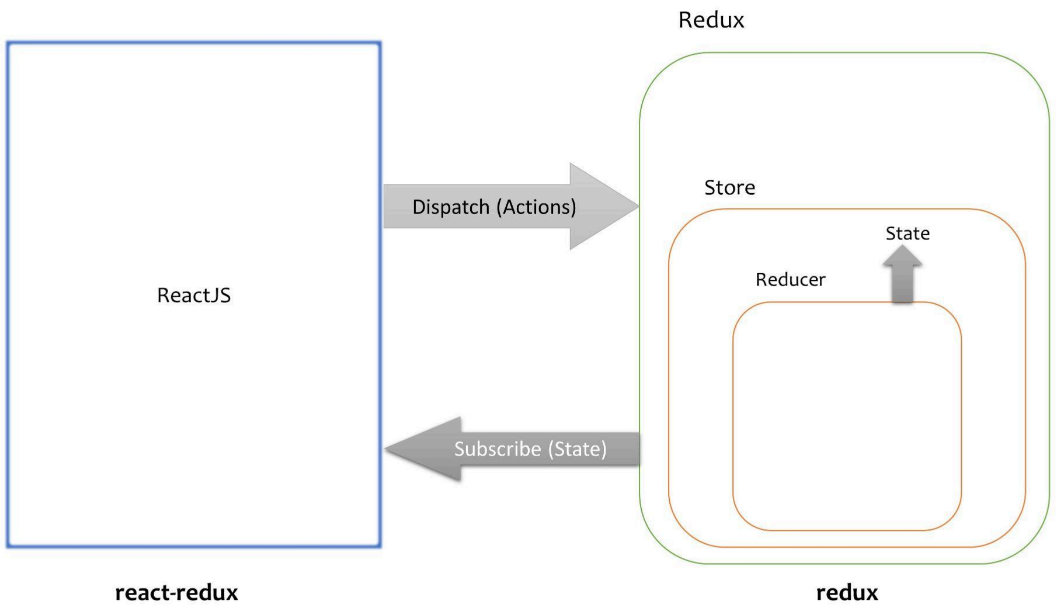
https://shorturl.at/3lCHe 15 Jun 2024

React With Redux
Multiple Reducers
Use of Thunk Middleware
Deploying React Application on Firebase
useState hook

===
Redux
===
- Redux is used for global state management.



Redux Architecture
- Create a store using 'redux' library.
- global business logic written in reducer.
- Output of the reducer is state.
- integrate this architecture with any front-end technology, eg ReactJS.
- 'react-redux' library is used to integrate react with redux.
- a request sent by reactjs is called as dispatch.
- dispatch contain various actions.
- Eg FETCH,
- WITHDRAW,
- UPDATE,
- DELETE,
- DEPOSIT,
- ...
- response received by reactjs is as subscribe
- subscribe contains state, implies received response is state.

Download libraries

redux

react-redux

```
>yarn add redux react-redux --save
```

Directory structure

```
<>
        src
                reduxeg
                        reducer
                                - reducer.js
                        - myComponent.js
        - index.js
```

create reducer

reducer.js

```
const initialState = {
    products: []
}

const reducer = (state = initialState, actions) => {
    switch (actions.type) {
        case 'PRODUCTS':
            return {
                ...state,
                products: [
                    { "p_id": 111, "p_name": "P_one", "p_cost": 10000 },
                    { "p_id": 222, "p_name": "P_two", "p_cost": 20000 },
                    { "p_id": 333, "p_name": "P_three", "p_cost": 30000 },
                    { "p_id": 444, "p_name": "P_four", "p_cost": 40000 },
                    { "p_id": 555, "p_name": "P_five", "p_cost": 50000 }
                ]
            }
    }
    return state
}
export default reducer
```

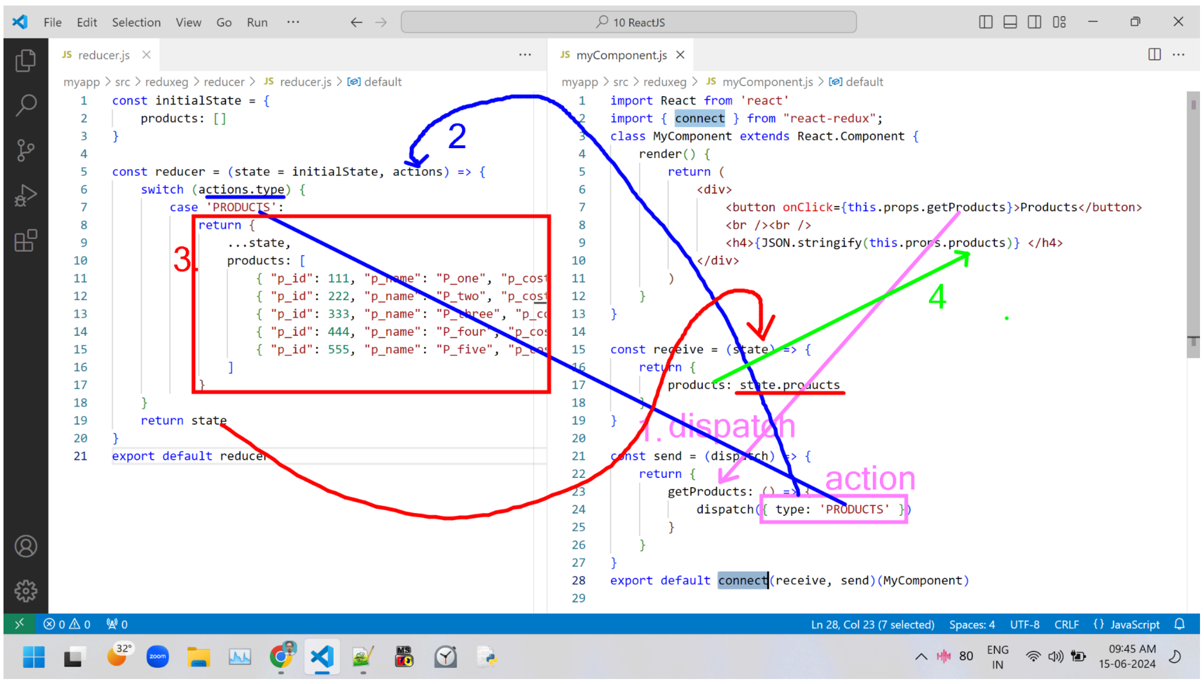
myComponent.js

```
import React from 'react'
import { connect } from "react-redux";
class MyComponent extends React.Component {
    render() {
        return (
            <div>
                <button onClick={this.props.getProducts}>Products</button>
                <br /><br />
                <h4>{JSON.stringify(this.props.products)} </h4>
            </div>
        )
    }
}

const receive = (state) => {
    return {
        products: state.products
    }
}

const send = (dispatch) => {
    return {
        getProducts: () => {
            dispatch({ type: 'PRODUCTS' })
        }
    }
}

export default connect(receive, send)(MyComponent)
```



File Edit Selection View Go Run
10 ReactJS
reducer.js
myapp > src > reduxeg > reducer > reducer.js > default
myComponent.js
myapp > src > reduxeg > myComponent.js > default
1. dispatch
2
3
4
action
Ln 28, Col 23 (7 selected) Spaces: 4 UTF-8 CRLF JavaScript
09:45 AM 15-06-2024

```
const initialState = {
    products: []
}

const reducer = (state = initialState, actions) => {
    switch (actions.type) {
        case 'PRODUCTS':
            return {
                ...state,
                products: [
                    { "p_id": 111, "p_name": "P_one", "p_cost
                    { "p_id": 222, "p_name": "P_two", "p_cost
                    { "p_id": 333, "p_name": "P_three", "p_co
                    { "p_id": 444, "p_name": "P_four", "p_cos
                    { "p_id": 555, "p_name": "P_five", "p_cos
                ]
            }
    }
    return state
}
export default reducer
```

```
import React from 'react'
import { connect } from "react-redux";
class MyComponent extends React.Component {
    render() {
        return (
            <div>
                <button onClick={this.props.getProducts}>Products</button>
                <br /><br />
                <h4>{JSON.stringify(this.props.products)} </h4>
            </div>
        )
    }
}

const receive = (state) => {
    return {
        products: state.products
    }
}

const send = (dispatch) => {
    return {
        getProducts: () => {
            dispatch({ type: 'PRODUCTS' })
        }
    }
}
export default connect(receive, send)(MyComponent)
```

index.js

```
//import reducer
import reducer from './06 reduxeg/reducer/reducer'
//import createStore
import { legacy_createStore as createStore } from 'redux';
//import Provider
import { Provider } from 'react-redux';
//create the store
const store = createStore(reducer)
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <MyComponent />
  </Provider>
);
```

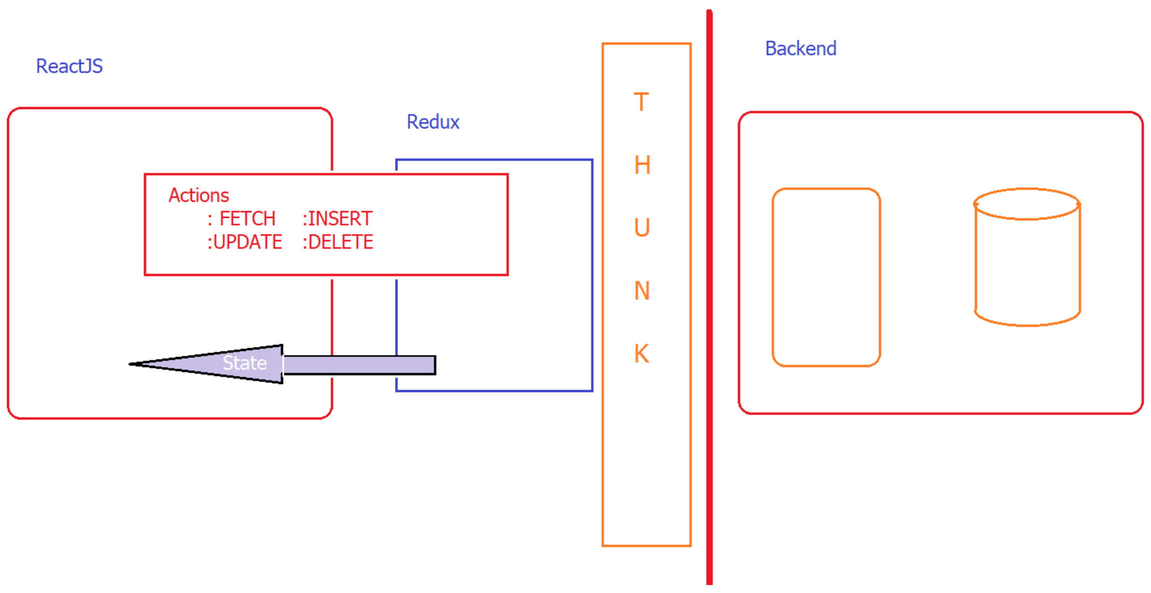
https://us06web.zoom.us/rec/play/FXteVH_Ym5B7SRD4vUEY7Lmc3XnzjEboR38qyg9OaJG8_zjhkZf1bo803D87Pj3dmazjn03hYhjHqoh1.lzZIAO5EISnTkE1a
Passcode: 9mqG*6&Z

==
Thunk Middleware
CRUD Application
==
1. Create react application

```
>create-react-app thunkapp
```

2. Switch to application

```
>cd thunkapp
```



Terminologies
- Actions
    : Used to monitor following actions
        : FETCH
        : INSERT
        : UPDATE
        : DELETE

- reducer (Redux)
    : used to maintain global states.

3. Download following libraries

- For api calls	-> axios
- For redux	-> redux
- For react with redux	-> react-redux
- Redux with thunk	-> redux-thunk
- Bootstrap styling	-> bootstrap, react-bootstrap

```
>yarn add axios redux react-redux bootstrap react-bootstrap
redux-thunk --save
```

4. Create actions

```
    <>
        src
            actions
                - actions.js
            - url.js
```

url.js

```
let url = `-- your url --`
export default url
```

actions.js

```
import axios from "axios"
import url from "../url"
const readAction = (records) => {
    return {
        type: 'FETCH', value: records
    }
}

export const getProducts = () => {
    return (dispatch) => {
        return axios.get(url + '/fetch')
            .then((posRes) => {
                dispatch(readAction(posRes.data))
            }, (errRes) => {
                console.log(errRes)
            })
    }
}
```

5. Create reducer

```
<>
    src
        reducer
            - reducer.js
```

reducer.js

```
const intialState = {
    data: []
}

const reducer = (state = intialState, actions) => {
    switch (actions.type) {
        case 'FETCH':
            state.data = []
            return {
                ...state,
                data: state.data.concat(actions.value)
            }
    }
    return state
}
export default reducer
```

App.js

```
import React from 'react'
import * as actions from './actions/actions'
import { connect } from 'react-redux'
class App extends React.Component {
  componentDidMount() {
    this.props.getProducts()
  }
  render() {
    return (
      <div>
        data : {JSON.stringify(this.props.data)}
      </div>
    )
  }
}
const receive = (state) => {
  return {
    data: state.data
  }
}

const send = (dispatch) => {
  return {
    getProducts: () => {
      dispatch(actions.getProducts())
    }
  }
}

export default connect(receive, send)(App)
```

index.js

```
import { legacy_createStore as createStore } from 'redux';
import reducer from './reducer/reducer';
import { applyMiddleware } from 'redux';
import {thunk} from 'redux-thunk'
import { Provider } from 'react-redux';
```

```
const store = createStore(reducer, applyMiddleware(thunk))
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

```
//////////////////
      TEST APPLICATION AT THIS STAGE
      AFTER THAT PROCEED
//////////////////
```

actions.js

```
import axios from "axios"
import url from "../url"
const readAction = (records) => {
    return {
        type: 'FETCH', value: records
    }
}

export const getProducts = () => {
    return (dispatch) => {
        return axios.get(url + '/fetch')
            .then((posRes) => {
                dispatch(readAction(posRes.data))
            }, (errRes) => {
                console.log(errRes)
            })
    }
}

const insertAction = (result) => {
    return {
        type: 'INSERT', value: result
    }
}

export const insertProduct = (record) => {
    return (dispatch) => {
        return axios.post(url + '/insert', record)
            .then((posRes) => {
                dispatch(insertAction(posRes.data))
            }, (errRes) => {
                console.log(errRes)
            })
    }
}

const updateAction = (result) => {
    return {
        type: 'UPDATE', value: result
    }
}

export const updateProduct = (record) => {
    return (dispatch) => {
        return axios.post(url + '/update', record)
            .then((posRes) => {
                dispatch(updateAction(posRes.data))
            }, (errRes) => {
                console.log(errRes)
            })
    }
}

const deleteAction = (result) => {
    return {
        type: 'DELETE', value: result
    }
}

export const deleteProduct = (record) => {
    return (dispatch) => {
        return axios.post(url + '/delete', record)
            .then((posRes) => {
                dispatch(deleteAction(posRes.data))
            }, (errRes) => {
                console.log(errRes)
            })
    }
}
```

reducer.js

```
const intialState = {
    data: []
}

const reducer = (state = intialState, actions) => {
    switch (actions.type) {
        case 'FETCH':
            state.data = []
            return {
                ...state,
                data: state.data.concat(actions.value)
            }
        case 'INSERT':
        case 'UPDATE':
        case 'DELETE':
            return {
                ...state,
                state: actions.value
            }
    }
    return state
}
export default reducer
```

App.js
```
import React from 'react'
import * as actions from './actions/actions'
import { connect } from 'react-redux'
import { Modal, Table } from 'react-bootstrap'
import 'bootstrap/dist/css/bootstrap.css'
let arr = []
class App extends React.Component {
    constructor() {
        super()
        this.state = {
            loading: false,
            status: false,
            insertPopup: false,
            updatePopup: false
        }
    }
    showPopup = (msg) => {
        if (msg === `addRec`) {
            this.setState({
                status: true,
                insertPopup: true,
                updatePopup: false
            })
        }
        else {
            this.setState({
                status: true,
                insertPopup: false,
                updatePopup: true
            })
        }
    }
    closePopup = () => {
        this.setState({
            status: false
        })
    }
    componentDidMount() {
        if (arr != [])
            this.setState({
                loading: true
            })
        else
            this.setState({
                loading: false
            })
        this.props.getProducts()
    }
    save = (e) => {
        e.preventDefault()
        if (this.state.insertPopup)
            this.insert(e)
        else if (this.state.updatePopup)
            this.update(e)
        this.closePopup()
    }
    insert = (e) => {
        let obj = {
            "p_id": e.target.p_id.value,
            "p_name": e.target.p_name.value,
            "p_cost": e.target.p_cost.value
        }
        this.props.insertProduct(obj)
        this.setState({
            result: "Insert Success"
        })
        arr.push(obj)
    }
    update = (e) => {
        let obj = {
            "p_id": e.target.p_id.value,
            "p_name": e.target.p_name.value,
            "p_cost": e.target.p_cost.value
        }
        this.props.updateProduct(obj)
        this.setState({
            result: "Update Success"
        })
        arr.forEach((e) => {
            if (e.p_id == obj.p_id) {
                e.p_name = obj.p_name
                e.p_cost = obj.p_cost
            }
        })
    }
    delette = (_id) => {
        this.props.deleteProduct(_id)
        this.setState({
            result: "Delete Success"
        })
        arr.splice(arr.findIndex((e, i) => {
            return e.p_id == _id
        }), 1)
    }
    render() {
        arr = this.props.data
        return (
            <div className='container mt-5'>
                <button className='btn btn-outline-primary mb-2 mr-auto'
                    onClick={() => { this.showPopup('addRec') }}>
                    Add +
                </button>
                {/* -----  modal code start------- */}
                <Modal show={this.state.status}
                    onHide={this.closePopup}
                    size='sm'
                    centered>
                    <div className='modal-header'>
```

```
                        <div className='modal-title'>Add / Update</div>
                    </div>
                    <div className='modal-body'>
                        <form onSubmit={this.save}>
                            <div className='form-group'>
                                <label>P_ID</label>
                                <input type='number'
                                    className='form-control my-2'
                                    placeholder='Enter P_ID'
                                    name='p_id'></input>
                            </div>

                            <div className='form-group'>
                                <label>P_NAME</label>
                                <input type='text'
                                    className='form-control my-2'
                                    placeholder='Enter P_NAME'
                                    name='p_name'></input>
                            </div>

                            <div className='form-group'>
                                <label>P_COST</label>
                                <input type='number'
                                    className='form-control my-2'
                                    placeholder='Enter P_COST'
                                    name='p_cost'></input>
                            </div>
                            <input type='submit' value='Add / Update'
className='btn btn-success m-3'></input>
                            <button className='btn btn-danger m-3'
onClick={this.closePopup}>Close</button>
                        </form>
                    </div>
                </Modal>
                {/* -----  table code start------- */}
                <Table bordered
                    variant='primary'
                    size='sm'
                    hover
                    striped
                    className='text-center'>
                    <thead>
                        <tr>
                            <th>SNO</th>
                            <th>ID</th>
                            <th>NAME</th>
                            <th>COST</th>
                            <th>EDIT</th>
                            <th>DELETE</th>
                        </tr>
                    </thead>
                    <tbody>
                        {arr.map((element, index) => (
                            <tr key={index}>
                                <td>{index + 1}</td>
                                <td>{element.p_id}</td>
                                <td>{element.p_name}</td>
                                <td>{element.p_cost} </td>
                                <td><button className='btn btn-warning'
onClick={() => { this.showPopup("update") }}> E </button> </td>
                                <td><button className='btn btn-outline-danger'
onClick={() => { this.delette(element.p_id) }}> D </button> </td>
                            </tr>
                        ))}
                    </tbody>
                </Table>
                <h1 className='text-info'>{this.state.result} </h1>
            </div>
        )
    }
}
const receive = (state) => {
    return {
        data: state.data
    }
}
const send = (dispatch) => {
    return {
        getProducts: () => { dispatch(actions.getProducts()) },
        insertProduct: (record) => { dispatch(actions.insertProduct(record))
},
        updateProduct: (record) => { dispatch(actions.updateProduct(record))
},
        deleteProduct: (id) => { dispatch(actions.deleteProduct({ "p_id": id
})) }
    }
}
export default connect(receive, send)(App)
```

Deploying react application to (Firebase)

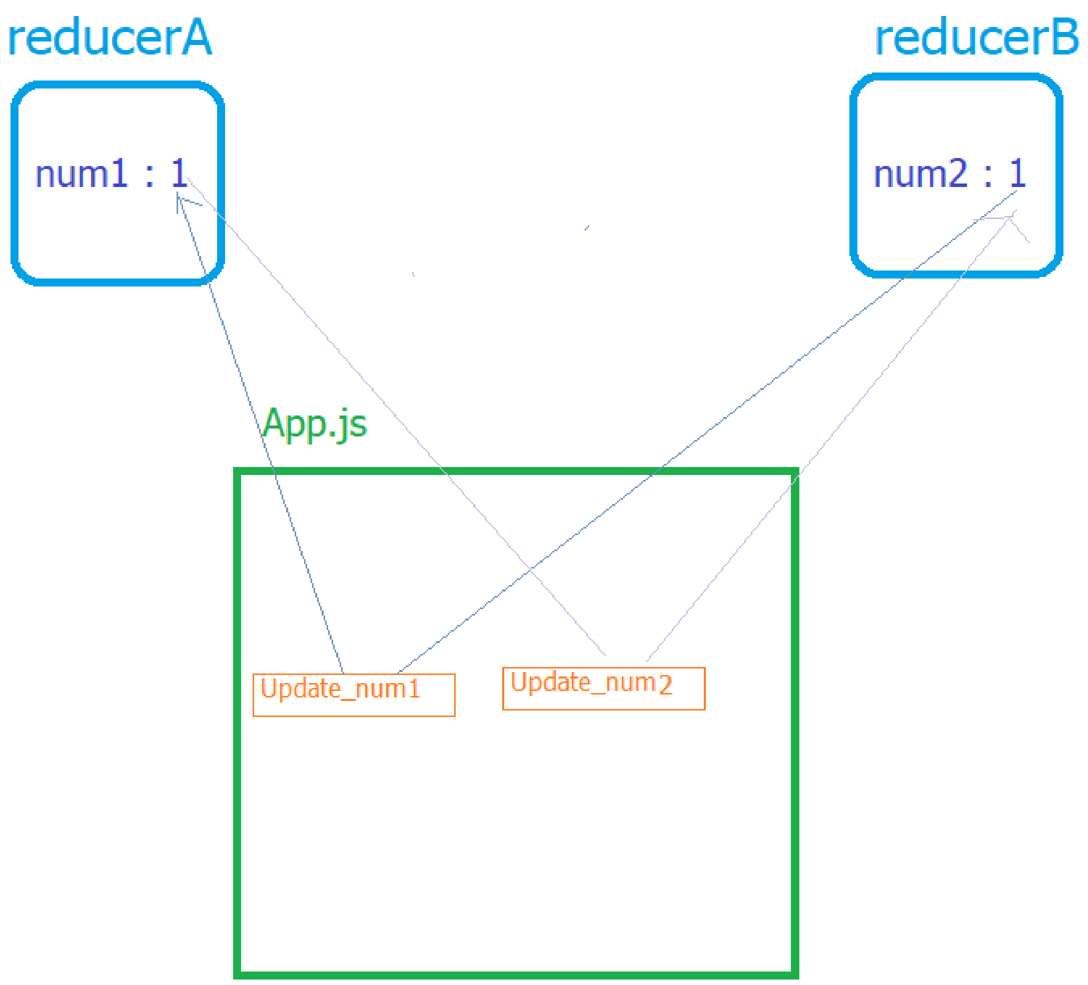
* build ReactJS application
    >npm run build

1. https://console.firebase.google.com/
2. Create new project
3. Continue 2 times
4. Configure Google Analytics -> default account
5. Click on create project wait till finish setup
    click on continue
6. Click on web (</>)
    register app
    add firebase sdk
    left side panel under build select hosting
7. click on get started
8. install firebase tools
    >npm install -g firebase-tools
9. after installing click on next
10. Initialyse your project
    Sign in to google
        >firebase login

11. initialyse project
    >firebase init
    - y
    - select hosting Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
    - hit spacebar to select and hit enter
    - use existing project -> select projectname
    - public directory 'build'
    - configure single page application -> y
    - setup auto deploy -> no
    - **DO NOT OVERWRITE index.html**
12. Click on next
13. Firebase deploy
    >firebase deploy
14. Click on continue to console

https://us06web.zoom.us/rec/play/UDy7w-ZwUKtj9D7E26X1DnaXAiouxxvEOM2p6dTXCkC4gb9u45IP_4R7ChXQwQ5h19GK5Snc5HTJsfI5.4kOcdlYDFfBuf8NR
Passcode: w!.0C???

==
Multiple Reducers
==



<>
src

```
        Multireducers
            - reducerA.js
            - reducerB.js
        - app.js
        - index.js
```

reducerA.js

```
const initialState = {
    num1: 1    //num2 : 1
}
const reducerA = (state = initialState, actions) => {
    switch (actions.type) {
        case 'UPDATE_A':    //'UPDATE_B'
```

```
            return {
                ...state,
                num1: state.num1 + actions.value
                //num2 : state.num2 + actions.value
            }
    }
    return state
}
export default reducerA //reducerB
```

similarly design reducerB

App.js

```
import React from 'react'
import { connect } from "react-redux";
class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Num1:- <span style={{ color: 'red', margin: '100px'
}}>{this.props.num1} </span> </h1>
        <h1>Num2:- <span style={{ color: 'red', margin: '100px'
}}>{this.props.num2} </span> </h1>
        <br /><br />
        <button style={{ margin: '100px' }} onClick={() => {
          this.props.updateNum1(this.props.num2)
        }}>Update_num1</button>
        <button style={{ margin: '100px' }} onClick={() => {
          this.props.updateNum2(this.props.num1)
        }}>Update_num2</button>
      </div>
    )
  }
}
const receive = (state) => {
  return {
    num1: state.rA.num1,
    num2: state.rB.num2
  }
}
const send = (dispatch) => {
  return {
    updateNum1: (data) => {
      dispatch({ type: 'UPDATE_A', value: data })
    },
    updateNum2: (data) => {
      dispatch({ type: 'UPDATE_B', value: data })
    }
  }
}
export default connect(receive, send)(App)
```

index.js

```
//import reducers
```

```
import reducerA from './Multireducers/reducerA'
import reducerB from './Multireducers/reducerB'
//import createStore
import { legacy_createStore as createStore, combineReducers } from 'redux';
//import Provider
import { Provider } from 'react-redux';
const rootReducer = combineReducers({
  rA: reducerA,
  rB: reducerB
})
const store = createStore(rootReducer)
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

==
Hooks
==

- Hook is an advanced part in ReactJS.
- It allows us to use advanced react features without writing a class.
- It has simplicity as good as functional components and features as good as class components.
- Hooks are introduced in February 2019 v16.8
- To work with hooks we must use functional components.
- Hooks wont work in class components.
- Hooks make code more readable.

Rule :-
- Hooks are declared before their use, implies declare at the top.
- Hooks are declared inside components.

Create react application
```
        >create-react-app hookapp
```
switch to application
```
        >cd hookapp
```
download bootstrap
```
        >yarn add bootstrap --save
```
execute application
```
        >yarn start
```

==
useState hook
==

- state is not supported by functional components.
- to work with state useState hook is used.

Eg01
- create a class component

- declare state variable count = 0
- on button click increment and decrement values

```
import React from 'react'
import 'bootstrap/dist/css/bootstrap.css'
export default class Eg01 extends React.Component {
    constructor() {
        super()
        this.state = {
            count: 0
        }
    }
    render() {
        return (
            <div>
                <button className='btn btn-outline-primary p-3'
onClick={this.dec}> - </button>
                <button className='btn btn-success mx-2
btn-lg'>{this.state.count} </button>
                <button className='btn btn-outline-primary p-3'
onClick={this.inc}> + </button>
            </div>
        )
    }
    inc = () => {
        this.setState({
            count: this.state.count + 1
        })
    }
    dec = () => {
        this.setState({
            count: this.state.count - 1
        })
    }
}
```

- implement the same using functional components

```
import { useState } from 'react'
export default function Eg01() {
    const [count, setCount] = useState(0)
    return (
        <div>
            <h1 className="text-primary">{count} </h1>
            <button className='btn btn-outline-info p-3 m-3' onClick={() =>
setCount(count - 1)}> - </button>
            <button className='btn btn-outline-info p-3 m-3' onClick={() =>
setCount(count + 1)}> + </button>
        </div>
    )
}
```

useState with objects

- Spot the issue

```
import { useState } from 'react'
export default function Eg02() {
    const [name, setName] = useState({
        fname: 'Fname',
        lname: 'Lname'
    })
    return (
        <div>
            <input type='text' placeholder='Enter First name' className='m-3'
                onChange={e => setName({ fname: e.target.value })} />
            <input type='text' placeholder='Enter Last name' className='m-3'
                onChange={e => setName({ lname: e.target.value })} />
            <h3>First Name:- {name.fname}</h3>
            <h3>Last Name:- {name.lname}</h3>
        </div>
    )
}
```

- issue is the 'useState' hook also can not preserve previous state.
- so do it manually using spread operator(...), as

```
      onChange={e => setName({ ...name, fname: e.target.value })} />
```

AND

```
      onChange={e => setName({ ...name, lname: e.target.value })} />
```

Eg03
useState with arrays

```
import { useState } from "react";
export default function Eg03() {
    const [times, setTimes] = useState([])
    let lap = () => {
        let today = new Date()
        setTimes([
            ...times,
            {
                id: times.length,
                value: today.getMinutes() + " : " + today.getSeconds() + " : "
+ today.getMilliseconds()
            }
        ])
    }
    return (
        <div>
            <button className="btn btn-outline-warning"
onClick={lap}><b>LAP</b></button>
            <ol>
                {
                    times.map(item => (
                        <li key={item.id}>{item.value}</li>
                    ))
                }
            </ol>
        </div>
    )
}
```

Conclusion:-

- This hook lets us add state in functional components.
- In the class component, state must be an object.
- With useState hook, state must not be an object.
- The useState hook returns an array of two elements.
- First element is the current value of state.
- Second element is the state setter method.

Attendance link for 16 June 2024

https://bit.ly/HH-150624