Redux Thunk

Redux is a popular and open-source [Javascript](https://www.javatpoint.com/javascript-tutorial)

library primarily aimed to cover the management measure states of the application. It is also commonly used to develop user interactive interfaces with React or Angular.

Additionally, it is quite absolute to know that the actions of Redux are dispatched **asynchronously**. It is quite a problem for non-trivial applications that need connection with Application Program Interface or API to perform external effects side by side. Redux can also be considered as a **middleware** that deals with dispatching and reaching between actions and reducers. Since Redux's actions are asynchronously dispatched, two middleware libraries serve the purpose. They are **Redux-Saga** and **Redux Thunk**. Here, we will discuss Redux Thunk.

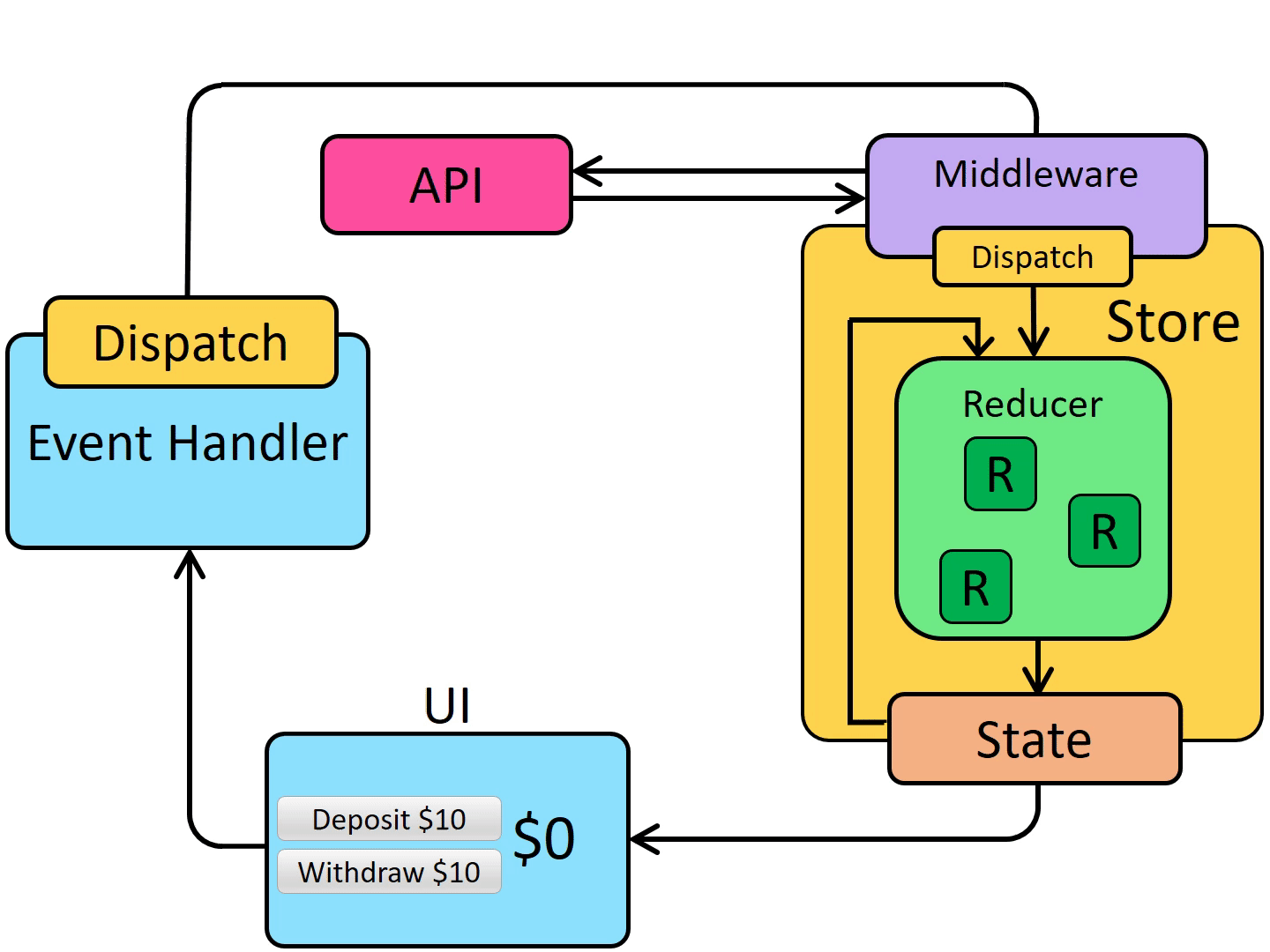
What is Redux Thunk?

Thunk is a logical concept in programming where you deal with a function that is primarily used to delay the calculation or evaluation of any operation.

Redux Thunk acts as a middleware that will return you a function instead of an object while calling through the action creators. The returned function receives the dispatch method from the store and then later it is used to dispatch synchronously inside the body of function once the asynchronous actions have been completed. Let's move ahead with Redux Thunk and talk more about its installation and usage.

The main objective of using Redux Thunk is to establish asynchronous communication from the externally present API into fetching and saving data. It makes it easier to dispatch actions that follow the paradigm of the whole lifecycle of requests from the externally present API.

 It is important to keep in mind that Redux allows you to synchronously dispatch actions in our React application while creating and utilizing store and depending on the external API for close call integration of data.



**stores.js**

import counterReducer from "./counterReducer";

import { combineReducers, createStore, applyMiddleware } from "redux";

import todoReaducer from "./todoReducer";

import productReducer from "./productReducer";

import thunk from 'redux-thunk'

var **reducer** = **combineReducers**({**counter**:**counterReducer**,**todolist**:**todoReaducer**,**products**:**productReducer**})

var mainstore = new **createStore**(**reducer**,**applyMiddleware**(thunk))

export default mainstore;

**Product reducer.js**

const initialState = {

    products:[],

    isLoading:false,

    cart:[]

}

function **productReducer**(state=initialState,action){

    if(action.type==='UPDATE\_PRODUCTS'){

        return {...state,products:[...action.payload]}

    }

    if(action.type==='TOGGLE\_LOADER'){

        return {...state,isLoading:!state.isLoading}

    }

    if(action.type==='ADDTOCART'){

        action.payload.count=1

        return {...state,cart:[...state.cart,action.payload]}

    }

    if(action.type==='INC-CART-PROD-COUNT'){

        var temp = [...state.cart];

        temp = temp.**map**((cp)=>{

            if(cp.id===action.payload.id){

                cp.count++

            }

            return cp

        })

        return {...state,cart:[...temp]}

    }

    if(action.type==='DEC-CART-PROD-COUNT'){

        temp = [...state.cart];

        temp = temp.**map**((cp)=>{

            if(cp.id===action.payload.id){

                cp.count--

            }

            return cp

        })

        return {...state,cart:[...temp]}

    }

    if(action.type==='DELETE-CART-PROD'){

        temp = [...state.cart];

        temp = temp.**filter**((cp)=>{

            if(cp.id===action.payload.id){

                return false

            }

            else{

                return true

            }

        })

        return {...state,cart:[...temp]}

    }

    return state

}

export default **productReducer**

**­­­**

**action.js**

export function **addProductToCart**(product){

    return {type:'ADDTOCART',payload:product}

}

export function **getAllProducts**(){

    return  (dispatch)=>{

**dispatch**({type:'TOGGLE\_LOADER'})

**fetch**("http://localhost:4000/products")

        .**then**(res=>res.**json**())

        .**then**(data=>{

            console.**log**(data)

**dispatch**({type:'TOGGLE\_LOADER'})

**dispatch**({type:'UPDATE\_PRODUCTS',payload:data})

        })

    }

}

**Products.jsx**

import React from 'react'

import { connect } from 'react-redux'

import { useNavigate } from 'react-router-dom'

import {addProductToCart, getAllProducts} from '../store/actions'

function **Products**({products:{products,cart,isLoading},dispatch}) {

    var **navigate** = **useNavigate**()

    console.**log**(products)

    function **addToCart**(p){

**dispatch**(**addProductToCart**(p))

    }

**React**.**useEffect**(()=>{

**dispatch**(**getAllProducts**())

    },[])

    function **isInCart**(p){

        var temp = cart.**filter**((cp,i)=>{

            if(cp.id===p.id){

                return true

            }

            else{

                return false

            }

        })

        return !(temp.length===0)

    }

    return (

        <div *className*='d-flex flex-wrap justify-content-around'>

            {isLoading &&(<div *class*="spinner-border"></div>)}

            {

                products.**map**((p,i)=>{

                    return (

                        <div *className*="card m-2 text-center" *style*={{width:'250px'}}>

                            <div *className*='text-center'>

                                <img *className*="card-img-top" *src*={p.image} *alt*="Card image" *style*={{width:'50%',height:'100px'}} />

                            </div>

                            <div *className*="card-body">

                                <b>{p.title}</b>

                            </div>

                            <div *className*='card-footer'>

                                <div *className*='d-flex justify-content-between'>

                                    <h3><i *class*="bi bi-star-fill"></i>{p.rating.rate}</h3>

                                    <h3>{p.price}</h3>

                                </div>

                                {**isInCart**(p) && (<button *className*='btn btn-warning btn-sm' *onClick*={()=>{**navigate**("/cart")}}>Goto Cart</button>)}

                                {!**isInCart**(p) && (<button *className*='btn btn-success btn-sm' *onClick*={()=>{**addToCart**(p)}}>Add To Cart</button>)}

                            </div>

                        </div>

                    )

                })

            }

        </div>

    )

}

export default **connect**(store=>store)(**Products**)