# Redux Toolkit

## Step-01 Installation-

প্রথমে আমাদের npm i @reduxjs/toolkit react-redux ইনষ্টল করতে হবে

File Structure

src ফোল্ডারের মধ্যে features ফোল্ডার বানাব এরপর features ফোল্ডারের মধ্যে যে নামের ফিচার বানাব সেটির একটি ফোল্ডার বানাব। কারণ আমার ফিচারস অনেক থাকতে পারে।

src->features->counters->countersSlice.js

## Step-02- counterSlice.js –

import { createSlice } from "@reduxjs/toolkit"

const initialState = [

    {

        id: 1,

        count: 0

    },

    {

        id: 2,

        count: 0

    }

]

const counterSlice = createSlice({

    name: "counters",  //যে নাম দিয়ে স্লাইস লিখব সেই নাম দিব

    initialState,

    reducers: {

        increment: (state, action) => {

            const counterIndex = state.findIndex(st => st.id === action.payload) //এইখানে কাউন্টার ইন্ডেক্স টা পাচ্ছি [0]= {id:1} [1]={id:2}

            state[counterIndex].count++                 //[0].count=0+1=1

        },

        decrement: (state, action) => {

            const counterIndex = state.findIndex(st => st.id === action.payload)

            state[counterIndex].count--

        }

    }

})

export default counterSlice.reducer;  //রিডিউসার লিখতে হবে কারণ create slice সব গুলা রিডিউসারকে কম্বাইন করে একটা রিডিউসার বানিয়ে ফেলবে।

export const {increment,decrement} = counterSlice.actions; // রিডিসারের মধ্যে আমরা যে ফাংশন গুলো লিখব সেগুলাই action হিসেবে পরিচিত হবে। এবং এইটা আমরা পরবর্তীতে যেকোন জায়গা থেকে কল দিতে পারি

স্লাইস লিখার পর store ম্যানেজ করতে হবে—

## Step-03 Store Management

Folder Structure হলো –

src -> app->store.js

import { configureStore } from "@reduxjs/toolkit";

import countersReducer from "../features/counters/coutersSlice"; // CounterSLice কে ইম্পোর্ট করলাম

const store = configureStore({

    reducer: {

        counters: countersReducer,

    }

})

export default store;

## Step-04- Wrapping with Redux Toolkit Provider

import { StrictMode } from "react";

import { createRoot } from "react-dom/client";

import { Provider } from "react-redux"; // react-redux থেকে ইম্পোর্ট করেছি

import store from "./app/store";

import Homepage from "./components/Homepage";

import "./index.css";

createRoot(document.getElementById("root")).render(

  <StrictMode>

    {/\* Provider দিয়ে wrap করে দিয়েছি।  \*/}

    <Provider store={store}>

      <Homepage />

    </Provider>

  </StrictMode>

);

## Step-05: Using useSelector to do in components

এই খানে আমরা slice এর state থেকে ডাটা নিয়ে আসার জন্য useSelector ব্যবহার করতেছি।

আর Action গুলো Dispatch করার জন্য useDispatch ব্যবহার করতেছি।

import React from "react";

import { useDispatch, useSelector } from "react-redux";

import { increment } from "../features/counters/coutersSlice";

const Homepage = () => {

  const counters = useSelector((state) => state.counters); // এইখানে state টি হলো store এ থাকা reducer এর মধ্যে থাকা object টি । এইজন্যই state.counters লিখা হয়েছে। আর এই useSelector ব্যবহৃত হয় slice এর initialState থেকে  ডাটা নিয়ে আসার জন্য ;

  const dispatch = useDispatch();

  console.log(counters);

  return (

    <div className="flex flex-col justify-center items-center">

      {counters.map((counter) => (

        <div

          key={counter.id}

          className="text-center space-y-6 mt-5 border border-gray-500 p-2"

        >

          <h1>{counter.count}</h1>

          <div className="flex gap-6">

            <button

              onClick={() => dispatch(increment(counter.id))} // dispatch এর মধ্য আমরা increment Action কে কল দিয়েছি। এর মধ্যে শুধু একটা ডাটাই পাঠাতে পারব আর সেটা যাবে action.payload আকারে। যদি আমাকে মাল্টিপল ডাটা পাঠাতে হয় তাহলে আমাকে Object আকারে পাঠাতে হবে।

              className="px-4 py-2 bg-green-500"

            >

              Increment

            </button>

            <button className="px-4 py-2 bg-red-500">Decrement</button>

          </div>

        </div>

      ))}

    </div>

  );

};

export default Homepage;

# Redux Query

## Step-01- Store Configuration

import { configureStore } from "@reduxjs/toolkit";

import { apiSlice } from "../features/api/apiSlice";

const store = configureStore({

    reducer: {

        [apiSlice.reducerPath]: apiSlice.reducer

    },

    middleware: (getDefaultMiddlewares) => getDefaultMiddlewares().concat(apiSlice.middleware),

})

export default store;

## Step-02: Create ApiSlice

folder Structure

features>api>apiSlice.js

এর ভেতরে নিচের কোড কপি পেষ্ট করতে হবে। মুলত আমরা এখানেই সব api লিখব তবে

[নোটঃ- import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react']

এইটা কপি পেষ্ট করতে হবে।

## Step-03: (GetAllData)

### ApiSlice

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'

export const apiSlice = createApi({

    reducerPath: "api",

    baseQuery: fetchBaseQuery({

        baseUrl: " http://localhost:9000"

    }),

    endpoints: (builder) => ({

        getVideos: builder.query({

            query: () => "/videos"

        })

    })

})

export const { useGetVideosQuery } = apiSlice

### Component

import React from "react";

import { useGetVideosQuery } from "../features/api/apiSlice";

import SingleVideo from "./SingleVideo";

const VideoBoard = () => {

  const { data: videos, isLoading, isError } = useGetVideosQuery();

  //what to render

  let content = null;

  if (isLoading) {

    content = <div>Loading...</div>;

  }

  if (!isLoading && isError) {

    content = <div>Error Occured</div>;

  }

  if (!isLoading && !isError && videos.length === 0) {

    content = <div>No Videos Found</div>;

  }

  if (!isLoading && !isError && videos.length > 0) {

    content = videos.map((video) => (

      <SingleVideo key={video.id} video={video} />

    ));

  }

  return (

    <section className="pt-6 pb-20 min-h-[calc(100vh\_-\_157px)]">

      <div className="grid grid-cols-12 gap-4 max-w-7xl mx-auto px-5 lg:px-0 min-h-[300px]">

        {content}

        {/\* <div className="col-span-12">some error happened</div> \*/}

      </div>

    </section>

  );

};

export default VideoBoard;

## Step-04: GetDataByID

### apiSlice

import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'

export const apiSlice = createApi({

    reducerPath: "api",

    baseQuery: fetchBaseQuery({

        baseUrl: " http://localhost:9000"

    }),

    endpoints: (builder) => ({

        getVideos: builder.query({

            query: () => "/videos"

        }),

        getVideo: builder.query({

            query: (videoId) => `/video/${videoId}`

        })

    })

})

export const { useGetVideosQuery, useGetVideoQuery } = apiSlice

### Component ---

import React from "react";

import { useParams } from "react-router-dom";

import { useGetVideoQuery } from "../features/api/apiSlice";

import Description from "./Description";

import Player from "./Player";

const VideoPlayer = () => {

  const { videoId } = useParams();

  console.log(videoId);

  const { data: video, isLoading, isError } = useGetVideoQuery(videoId);

  console.log(video);

  //what to render

  let content = null;

  if (isLoading) {

    content = <div>Loading...</div>;

  }

  if (!isLoading && isError) {

    content = <div>Error Occured</div>;

  }

  if (!isLoading && !isError && video?.id) {

    content = (

      <>

        <Player />

        <Description />

      </>

    );

  }

  return (

    <section class="pt-6 pb-20 min-h-[calc(100vh\_-\_157px)]">

      <div class="mx-auto max-w-7xl px-2 pb-20 min-h-[400px]">

        <div class="grid grid-cols-3 gap-2 lg:gap-8">

          <div class="col-span-full w-full space-y-8 lg:col-span-2">

            {content}

          </div>

          {/\* <!-- related videos --> \*/}

          <div class="col-span-full lg:col-auto max-h-[570px] overflow-y-auto">

            {/\* <!-- single related video --> \*/}

            <div class="w-full flex flex-row gap-2 mb-4">

              <div class="relative w-[168px] h-[94px] flex-none duration-300 hover:scale-[1.03]">

                <a href="video.html">

                  <img

                    src="https://i3.ytimg.com/vi/6O4s7v28nlw/maxresdefault.jpg"

                    class="object-cover"

                    alt="Some video title"

                  />

                </a>

                <p class="absolute right-2 bottom-2 bg-gray-900 text-gray-100 text-xs px-1 py">

                  12:10

                </p>

              </div>

              <div class="flex flex-col w-full">

                <a href="#">

                  <p class="text-slate-900 text-sm font-semibold">

                    Some video title

                  </p>

                </a>

                <a

                  class="text-gray-400 text-xs mt-2 hover:text-gray-600"

                  href="#"

                >

                  Learn with Sumit

                </a>

                <p class="text-gray-400 text-xs mt-1">

                  100K views . 23 Oct 2022

                </p>

              </div>

            </div>

          </div>

        </div>

      </div>

    </section>

  );

};

export default VideoPlayer;

## Step-05: POST Request

এইখানে আমরা builder এর mutation ব্যবহার করেছি কারণ এইটা POST

### apiSlice

  addVideo: builder.mutation({

            query: (data) => ({

                url: "/videos",

                method: "POST",

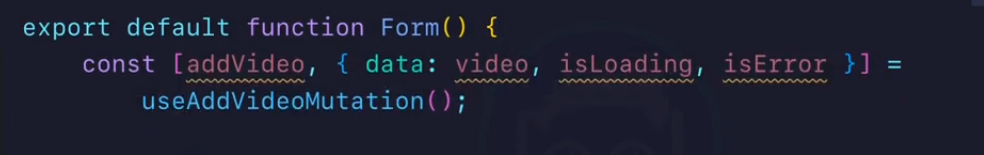
                body: data

            })

        })

export const {useAddVideoMutation } = apiSlice

### Components

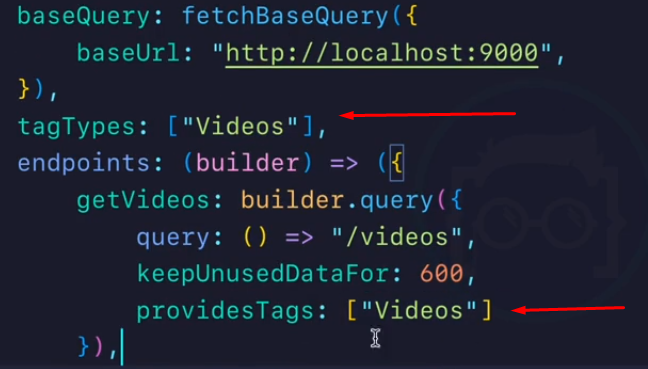


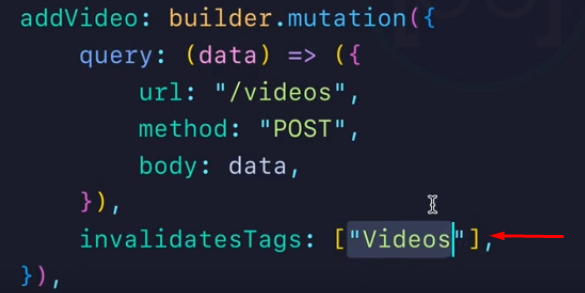
### Revalidation (importnat)

আমরা কোন একটি ভিডিও/পোষ্ট/ডাটা অ্যাড করলাম কোন ফরম দ্বারা এবং অ্যাড করার সাথে সাথে ওইটার ভিডিও লিষ্ট বা ডাটা লিষ্ট ক্যাশ ইনভেলিড করে আবার নতুন ডাটা নিয়ে আসবে এইজন্য আমরা revalidation করব।

এক্ষেত্রে আমাদের তিনটা জিনিষ যুক্ত করতে হবে।

1. tagTypes:[] – এইটা একটা অ্যারের যেইটা আমরা baseQuery পর লিখব। আর এইখানে যে নামগুলো থাকবে সেগুলাই হোয়াইট লিষ্ট । এই হোয়াইট লিষ্ট দিয়ে রিডাক্স সব কিছু কন্ট্রোল করবে।
2. provideTags:[] – এইটাও একটা অ্যারে । এখানেও আমরা মাল্টিপল ট্যাগ রেখে দিতে পারি। এইটা আমরা ব্যবহার করব যে কুয়েরী আমরা ইনভেলিডেট করে দিতে চাই অর্থাৎ অ্যাড ফর্ম করার পর যে লিষ্ট আপডেট হবে সেই কুয়েরীতে লিখব ।
3. invalidatesTags:[]—এইটাও একটা অ্যারে। যেখানে আমরা সাবমিট করব অর্থাৎ POST কুয়েরী যেখানে চালিয়েছি সেখানে আমরা invalidatesTags বলে দিব অ্যারে আকারে।

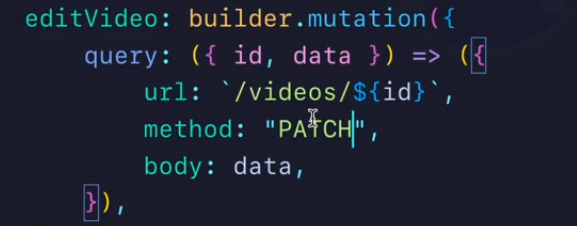




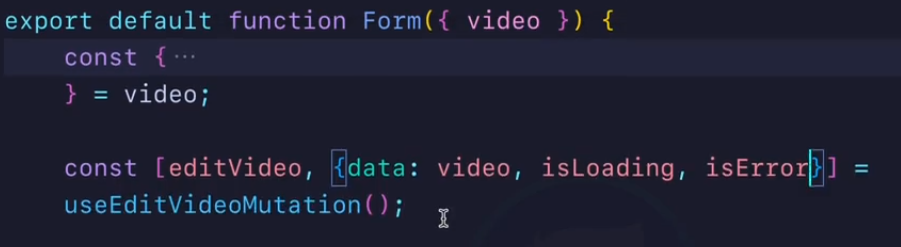
## Step-06: Edit Request

### apiSlice

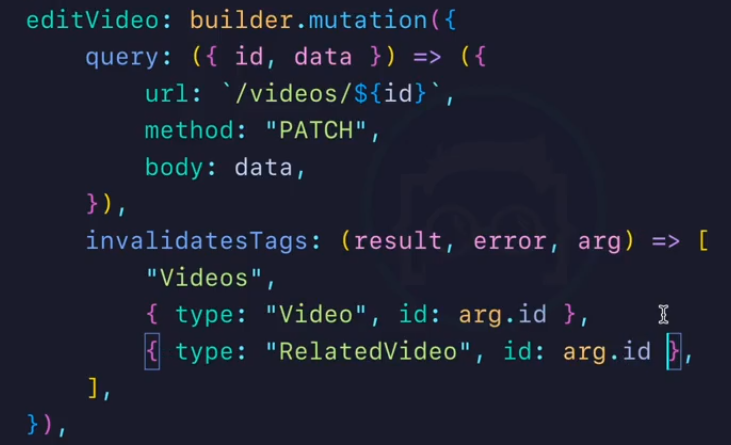
এইখানে আমরা query তে দুইটা প্যারামিটার নিব। {id,data}



### Component



### Revalidation (Each Post will be Invalid)



এইখানে invalidatesTags: এর মধ্যে তিনটি জিনিষ পেয়েছে।

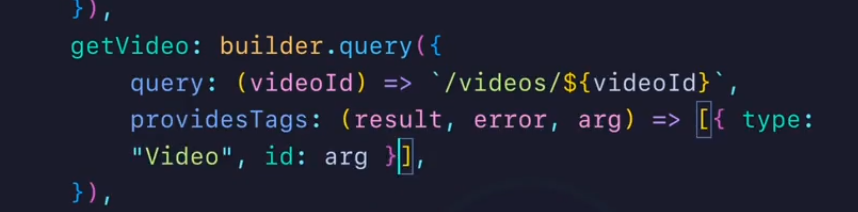
প্রথমটি হচ্ছে result । এইটা যখন আমি editVideo করে সাবমিট করে তারপর যে ডাটা পাব সেইটা result এ গিয়ে ষ্টোর হবে।

দ্বিতীয়টি error তো আমরা জানি যদি কোন ইরোর হয় তাহলে আমরা এখানে পাব।

তৃতীয়টি হলো arg যেইটা আমরা editVideo ফাংশনের কুয়েরীর মধ্যে পাব সেটি।

* উপরের কোডে প্রথমে Videos ট্যাগ টি ইনভেলিডেট হয়ে যাবে যেইটা আগের getVideos এ প্রোভাইড করা হয়েছিল। পরের লাইনে {type:”Video”,id:arg.id} তে এইটা প্রত্যেক টা ভিডিওর একটা অ্যারে তৈরি করবে । এবং type এর “Video” ট্যাগ টা প্রভাইড করা আছে getVideo তে । মানে যখন প্রত্যেকটা ভিডিও কল হচ্ছে তখন প্রত্যাকটা ভিডিওর একটি ক্যাশ তৈরি হচ্ছে সেখানে।

নিচে একটি কোড যুক্ত করা হলো।

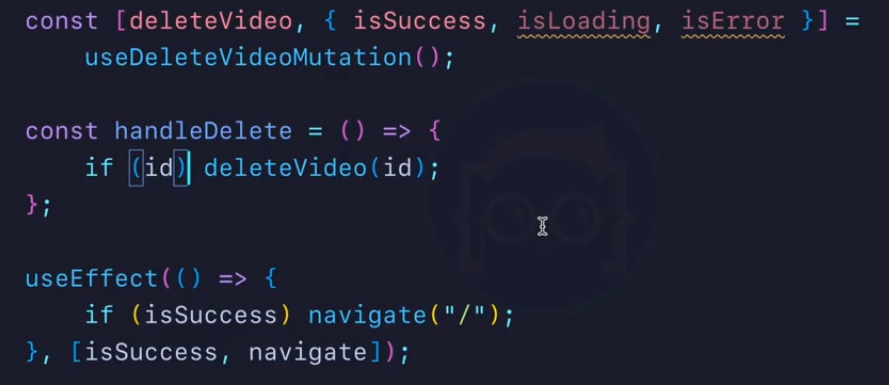


## Step-07: Delete Request

apiSlice



component



# Attributes

**References- https://redux-toolkit.js.org/rtk-query/usage/queries**

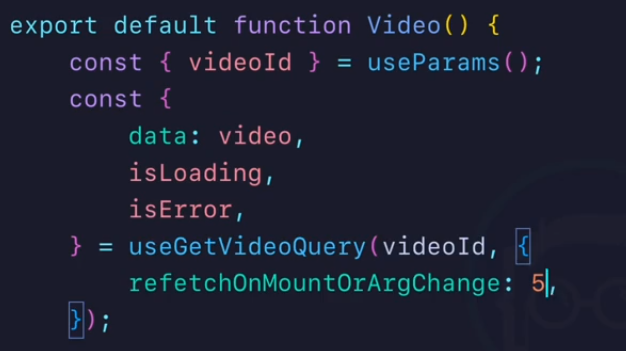
## **refetchOnFocus**

আমরা এই ট্যাবে থাকতেই আবার অন্য কোন ট্যাবে চলে যাই তাহলে এই কম্পোনেন্ট আনমাউন্ট হয়ে গেলো। এরপর আবার যখন অন্য ট্যাব হবে এই ট্যাবে ব্যাক করব তখন পুনরায় ডাটা ফেচ হবে। {এইজন্য যে জায়গায় কুয়েরী ব্যবহার করেছি সেই জায়গায় refetchOnFocus True করে দিতে হবে}



## refetchOnMountOrArgChange

আমরা ক্লিক দিয়ে একটি কম্পোনেন্টে ঢুকার পর আবার হোমে এসে আবার ওই একই কম্পোনেন্টে গেলে যদি প্রোপস চেঞ্জ হয়ে যায় তাহলে সে আবার ফেচ করবে।



## skip—

এইটা মাথ্যমে আমরা কোন কুয়েরী স্কিপ করে যেতে পারি।

## pollingInterval

এর মাধ্যমে আমরা মিলি সেকেন্ড পরপর পোলিং করতে থাকে।