FETCHING DATA IN REACT:

USE EFFECT:

Use effect is used to fetch data in React.

It is itself a function which takes arrow function and in that arrow function, we place call for the API.

So what happen is that we pass fetch part in useEffect arrow function under async function, why we use async is we know fetching data takes time, so fetching data will be done somewhere on server till then rest of the part of our react app loads and when fetching data is done, it will be show in the app. But the problem is the fetching takes time, so in the mean time what should we show, so for that we use useState and pass some value in it, so in mean time that value will be shown and when data got fetch, we update that value with the help of function in useState. Let see;

 let data = {title: "Waiting for fetching"};

  let [todo, setTodo] = useState(data);

  //1. We are fetchin data in a same way as we did in JS, but here we used useEffect

  useEffect(()=> {

    async function fetchData(){

      const response = await fetch("https://jsonplaceholder.typicode.com/todos/1");//fetch data, await use to get resolve response

      //above we got the data, and to convert to json, we need await response.json();

      console.log("response = ", response);

      const data2 = await response.json();//converting to json

      setTodo(data2);

      console.log("data", todo);

    }

    fetchData();

  });

  return (

    <div >

        Hello Fetch

        <span>Title: {todo.title}</span>

    </div>

  );

}

Now this is the code.

1. We first define some before fetching data that displays in mean time of fetchin

 let data = {title: "Waiting for fetching"};

1. Then to update this data with our fetched data, we need something that can update data value, so we used useState and pass data to it, so todo has data value and setTodo has the ability to update that data value with the later fetched data.

  let [todo, setTodo] = useState(data);

1. Now we call useEffect, so we can fetch data in React and update UI in the runtime with the fetch data, fetching is done at the backend, and after fetching, UI will be immediately updated with the fetch data, and before fetch data, we are showing data/todo which is “waiting for fetching”. Its like loading.

  useEffect(()=> {

    async function fetchData(){

      const response = await fetch("https://jsonplaceholder.typicode.com/todos/1");//fetch data, await use to get resolve response

      //above we got the data, and to convert to json, we need await response.json();

      console.log("response = ", response);

      const data2 = await response.json();//converting to json

      setTodo(data2);

      console.log("data", todo);

    }

    fetchData();

  });

* In useEffect function, we pass arrow function and in that arrow function we pass asynch function so that delay part or fetching done without crashing our app in the backend. Everything is same, change takes place at const data2, in that we store json data and than to update our prevous “Waiting for fetching” with the newly fetched data, we pass data2 in setTodo so it will update the value of todo with the latest fetched data, and to make sure all this process happens, we call fetchData() function in useEffect and thus our data got updated and we can use it in our real time UI which is in div or in components

   fetchData();

  });

  return (

    <div >

        Hello Fetch

        <span>Title: {todo.title}</span>

    </div>

  );

}

export default App

1. Here we in span, we use {todo.title}. So before fetching was done, todo which has data so todo.title->data.title has “waiting for fetching”, so initially our app will show this and soon after fetching done, it shows title of fetched data.

IMP: One thing to remember that useEffect is only used when there is a change in data, if data is static, then there is no need to change data, like in our case, data is changing. Previously it was “waiting for fetching” and later fetched data came, so if data was static and we don’t need to change it in runtime, we donot use useEffect.