6 . Exploring the World

Fetch data dynamically from API and populate(display data on page) our page dynamically

2 approach to fetching data from API (backend)

1. **Loads -> API -> Render**
2. **Loads -> Render -> API -> Render**

1. Loads -> API -> Render

As soon as page load we make api call suppose it will take 500 ms so wait for 500 ms after 500ms it will render the UI

Eg . when we open webpage suddenly don't see anything as soon as api responds Quickly shows lot of stuffs on the screen

2. Loads -> Render -> API -> Render

As soon as page loads we will quickly render it then will make API call and as Soon as API responds now we will re-render our application with new data

As soon as page load we made api call we will wait for data to come as soon data As data come we will render it

In React we always using 2nd approach because it's this give us better UX

In 1st approach for 5000ms our page kind of frozen (don't see on page) and after 500ms we suddenly see anything so that's poor UI experience

Instead in the 2nd approach we load the page and render whatever we can render the skeleton and then later on so quickly when you render the page we can see something on website at least skeleton and then slowly website loads it's better user experience

Implementing 2nd approach using useEffect Hook

useEffect

Takes 2 arguments
1. Arrow function ()=>{}
2. Dependency array []

When useEffect callback (Arrow) function call?
The callback function called after our component render

Syntax

```
        useEffect ( () =>
{
        Callback function
 } , []);
```

Here [] - defendant array

```
import RestaurantCard from "./RestaurantCard";
import resList from "../utils/mockData";
import {useEffect, useState} from "react";

const Body =() =>
{

const [ListOfRestaurants, setListOfRestaurants]=useState(resList)


useEffect(()=>{},[])


    return(
            <div className="body">
                <div className="filter">

                <button className="filter-btn" onClick={()=>
                { //filter logic here
                 FilterList = ListOfRestaurants.filter((res)=>
                 res.info.avgRating > 4
                    );
                    setListOfRestaurants(FilterList);
                 }} >Top Rated Restaurants</button>

                </div>
```

Useffect will be helpful when we have to implement 2nd approach

When will this useEffect callback function called ?
The callback function will be called after our component render

When the body function will render and as soon as render cycle finished it will just quickly call the callback function

```
useEffect(()=>{
    console.log("useEffect called")
},[])
```

So how this code will works
1. The body component rendered
2. Callback function will called
Then UseEffect called will printed on console

Fetch will return a promise or we can use async await

Recommended to use async await

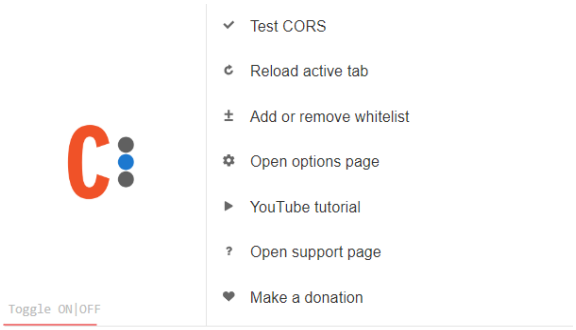
```
useEffect(()=>{
    fetchData();
},[])

const fetchData= async()=>
{
        const data= await fetch(
         "https://www.swiggy.com/dapi/restaurants/list/v5?
        );

       //convert Data into json
        const json = await data.json();
};
```

```
useEffect(()=>{
    fetchData();
},[])

const fetchData= async()=>
{
        const data= await fetch(
         "https://www.swiggy.com/dapi/restaurants/list/v5?lat
        );

       //convert Data into json
        const json = await data.json();

        //just seeing console swiggy api called or not
        console.log(json);
};
```

After run got this error because our browser

Our browser blockers to call API from one origin to diff origin
Our browser chrome is not allowing us to call SWIGGY's API from localhost
So for that
Add cors chrome extension

Then allow it by clicking on toggle on



```
const fetchData= async()=>
{
        const data= await fetch(
         "https://www.swiggy.com/dapi/restaurants/list/v5?lat=19.2183307&lng=72.9780897&is-seo-homepage-er
        );

       //convert Data into json
        const json = await data.json();

        //calling swiggy api
        console.log(json);

      setListOfRestaurants(json.data.cards[2].card.card.gridElements.infoWithStyle.restaurants);
};


   return(
            <div className="body">
                    <div className="filter">

                    <button className="filter-btn" onClick={()=>
                   { //filter logic here
```

So after refresh data will change

We can remove resList and mockData file

So now our whole app loading from Live data, it's making api and getting data or rendering data from the live API

It's not good way to write the code so we using optional chaining

```
const fetchData= async()=>
{
        const data= await fetch(
         "https://www.swiggy.com/dapi/restaurants/list/v5?lat=19.2183307&lng=72.9780897&is-seo-homepage-er
        );

       //convert Data into json
        const json = await data.json();

        //calling swiggy api
        console.log(json);

       //optional chainning
      setListOfRestaurants(json?.data?.cards[2]?.card?.card?.gridElements?.infoWithStyle?.restaurants);
};
```

When we refresh or open url its take some time to fetch data from API so unless we have data

spinning loader

We write the condition :-
If our ListOfRestaurant is empty then instead of rendering this all we will render
" Loading screen"

```
if (ListOfRestaurants.length===0)
{
    return <h1>Loading...</h1>
}
```

But this is not good way
So we Shimmer UI

Shimmer UI

It is kind of like we load fake page until we get actual data from api

> A shimmer UI resembles the page's actual UI, so users will understand how quickly the web or mobile app will load even before the content has shown up. It gives people an idea of what's about to come and what's happening (it's currently loading) when a page full of content/data takes more than 3 - 5 seconds to load.

So we creating separate component for that :- Shimmer

```
if (ListOfRestaurants.length===0)
{
    return <Shimmer/>;
}
```

instead of actual cards we showing shimmer UI
Here we wrote condition so it's known as condition rendering

Rendering basis on condition known as condition rendering

Here we can use conditional operator also

We creating new feature

By Default it will Login
When we click on the Login button it should change to Logout
And when we Click on Logout button it will change to Login button

We trying with normal javascript variable it will work ?

```
<div className="nav-items">
    <ul>
        <li>Home</li>
        <li>About Us</li>
        <li>Contact US</li>
        <li>Cart</li>
        <button className="login" onClick={()=>
        {
            btnName= "Logout";
            console.log(btnName);
        }}>{btnName}

        </button>
```

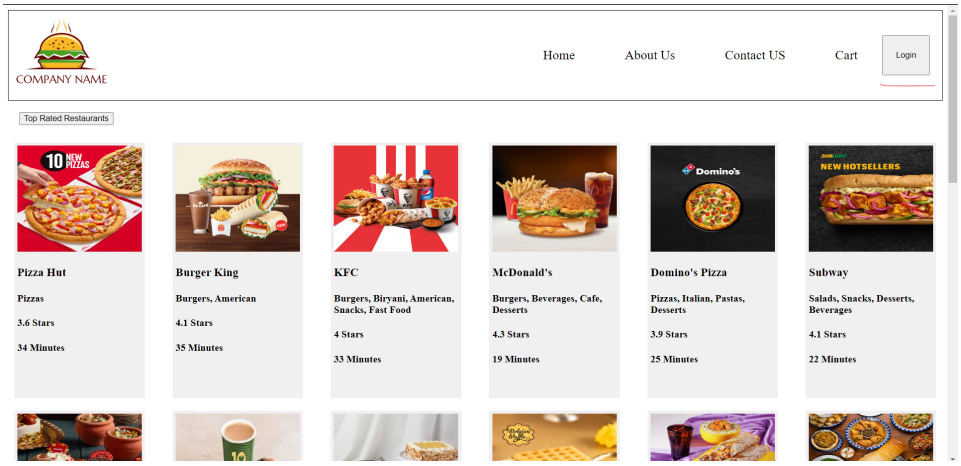
Means variable btnName got updated but UI did not updated

btnName got updated but our ui not rendered that is Header component did not get refreshed their would be some way that we can refreshed our Header component so it can take latest value of btnName this is why Javascript variable did not worked in such cases

If you want to make our component dynamic , if you want something should change in your component we use local state variable here use state variable comes into picture

```
<div className="nav-items">
    <ul>
        <li>Home</li>
        <li>About Us</li>
        <li>Contact US</li>
        <li>Cart</li>
        <button className="login" onClick={()=>
        {
            btnName==="Login"?
            setbtnName("Logout"):
            setbtnName("Login");

            console.log(btnName);
        }}>{btnName}

        </button>
```

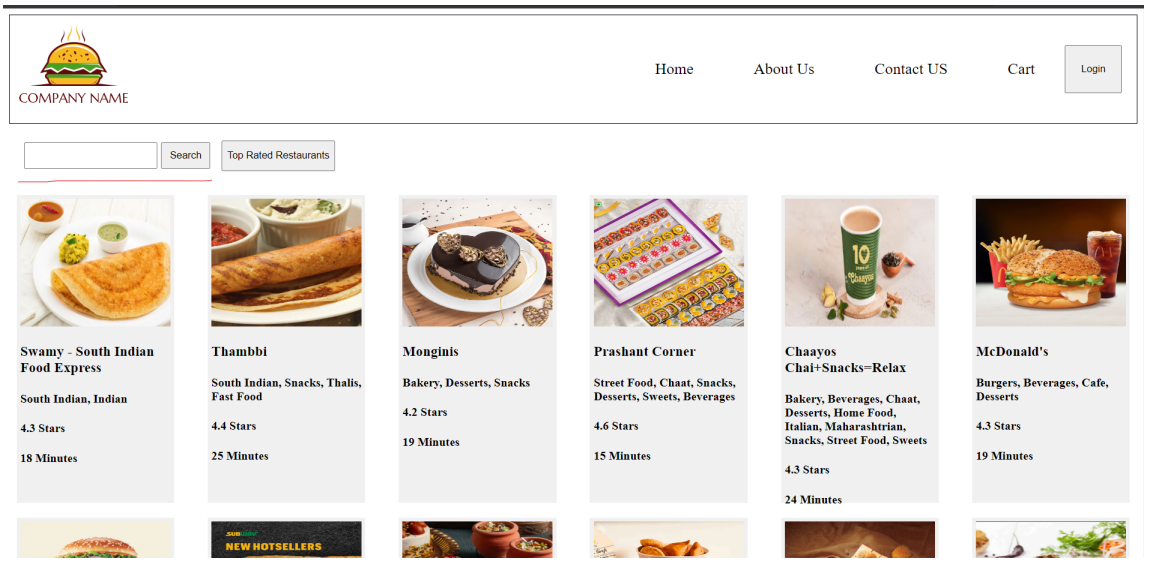
Using conditional operator



Here as soon as we call the setbtnName it will update the btnName value reference and it will render the Header component once again it will find the diff between older version and newer version and it will see in that div only button is getting updated their is no change in other HTML Elements only button is changed

That’s why react is best in DOM MAnipulation , it;s exactly know what to change so it only changes that
Whole component is rendered but only changes that we made change

Building this feature



```
<div className="search">
    <input type="text" className="search-box" value={}/>
    <button className="search-btn" onClick={()=>
    {
        //Filter the restarant cards and update the UI
        //SearchText
    }


    }>Search</button>
</div>
```

We will get search text from search box to get the data from input box we need to take the value of input box
we need dynamic data in search box so need to create local state variable

To track the value of input box whatever user is typing in to get that value we have to bind that value

```
<input type="text" className="search-box" value={searchText}
```

With the local state variable

```
let [searchText,setSearchText]= useState ("");
```

Then when we try to enter in search box but here it's not getting anything
Why search box is not working
Because whatever inside the searchtext state variable its inside the value of input box
When we changing the value of input box the value of input box is still is tide to the searchtext so our searchtext is not getting updated but we try to modify input box
It's can't happen because
The value of the value of searchtext (input box) is bind to the state variable of

searchtext and its empty so input box will not change unless we change the searchtext
So for fix this issue we use

Onchange handler here onchange handler also change searchtext (state variable)
How it's change - update with new value for that

```
onChange={ (e)=>
  {
      setSearchText(e.target.value)
} }/>
```

Here for update the searchtext with new value for getting new value (callback method e)

When we create input box first we need to bind with statevarible with input string (initial value)

Here when we typing search box we changing statevarible

Whenever we change local state variable React re-render the (Whole) component (when each key pressed react re-render the whole component)

React efficient dom manipulation
Efficient rendering

React fiber (react reconciliation algorithm) which find out the difference between two virtual DOM and update the DOM only when it required

//filter logic

```
<div className="search">
    <input type="text" className="search-box" value={searchText} onChange={(e)=>
    {
            setsearchText(e.target.value);

    }} />
    <button className="search-btn" onClick={()=>
    {
        //Filter the restarant cards and update the UI
        //SearchText

    const filteredRestarants= ListOfRestaurants.filter((res)=>
    res.info.name.toLowerCase().includes(searchText.toLowerCase())
    );
    setListOfRestaurants(filteredRestarants);

       // console.log(searchText);
    }
```

Problem

Whenever We doing filter one time we updated our ListOfRestaurants (suppose 2 res filtered)

And when we search once again we are searching in this 2 restaurants not the whole 15 restaurants , we lost the data of 15 restaurants which we got from API

So if we want search in our 15 restaurants it will not works

Solution

So for that we keep all the restaurants in the ListOfRestaurants and will create another copy of filtered restaurants

```
const Body =() =>
{

const [ListOfRestaurants, setListOfRestaurants]=useState([])
const [filteredRestarants, setfilteredRestarants]=useState([])

const [searchText, setsearchText]=useState("");

//console.log("Body Rendered");
```

Here initially filtered restaurants was empty

So whenever we doing filtered instead of updating ListOfRestaurants we will update FilteredRestarants

```
    <button className="search-btn" onClick={()=>
    {
        //Filter the restarant cards and update the UI
        //SearchText

    const filteredRestarants= ListOfRestaurants.filter((res)=>
    res.info.name.toLowerCase().includes(searchText.toLowerCase())
    );
    setFilteredRestarants(filteredRestarants);

       // console.log(searchText);
    }


    }>Search</button>
</div>
```

When we rendering we will rendered FilteredRestarants

```
{
FilteredRestarants.map((restaurant)=>
<RestaurantCard key={restaurant.info.id} resData={restaurant} />
)
}
```

So basically we kept all the restaurants in the ListOfRestaurants and we have FilteredRestarants so now whenever we applied filter to using ListOfRestaurants (never modified ListOfRestaurants just modified one time when we get data from API)

We set FilteredRestarants initially as empty so we need to put restaurants data in it

```
 //optional chainning
setListOfRestaurants(json?.data?.cards[2]?.card?.card?.gridElements?.infoWithStyle?.restaurants);
setFilteredRestarants(json?.data?.cards[2]?.card?.card?.gridElements?.infoWithStyle?.restaurants);
```

When we fetching data we updating data in ListOfRestaurants and copy of FilteredRestarants as well