Date.....

Chapter 05 - Let's get Hooked !

React File Structure

* React doesn't have opinions on how you put files into folders
* Some developers group their files by features.
* While moving files into folders are need to export it so that we can import it wherever necessary.

There are 2 ways of exporting :

(1) export default

→ This is the default way
→ This means you want to export only one value.

> A module is a self contained unit that can expose assets to other modules using exports, and acquire assets from other modules using import

→ There can be only one default export.
→ Default export is the value that will be imported from the module, if we use the simple import statement.

```
import Title from "./components/Title";
```

module

Title is the name that will be given locally to the variable assigned to contain the value, and it doesn't have to be named like the original export.

Date.....

ii) Named exports

Eg:- `export const Title = () => { }`

→ This is a named export with a name 'Title'
→ It can be imported like :-

`import { Title } from "./components/Header";`

→ If we created a new file which have 2 components and I want to export both of these components.

i) I can wrap these components into single object and can export.
ii) or I can export it seperately.

→ If Header.jsx have 2 components :-

Header & Title.

Then, either we can export each using names, or wrap it into single component and use default export. So ~~import~~

`import { Title, Header } from "/components/Header";`

or we can use

`import * as Obj from "./components/Header";`

Then we can use

'Obj.Title' in our code

Header.js

```
export const Title = () => {...}
const Header = () {...}
export default Header
```

App.js

```
import Header,
{Title} from "./compone/He
```


Teacher's Sign .....

Date.....

CONFIG FILE

or constant.js

→ Create a config file in your project.
→ I put all the hardcoded things into my config file (config.js)
→ Config file ~~can~~ will be like :-

```
const IMG_CDN_URL = "https://some url";
```

→ make this as named export

```
export const IMG_CDN_URL = "https--//";
```

→ Then import it like :-

```
import { IMG_CDN_URL } from "./config";
```

→ Put all hardcoded data. So put restaurantList also in config file.

Building Search Functionality

→ Search bar - inside Body
→ Const Body = () => {
return (
<>

<input
type = "text"
className = "search-input"
placeholder = "search"
value = ""

</>

Teacher's Sign .....

Date.....

```
<button className="search-btn">
    Search
</button>
</>
);
};
```

→ We've got search input and search button with us. But if I try to write inside my input box, it's not working (because it's controlled by React.

[If I write the same code inside my HTML file, it will work.]

ONE-WAY DATA BINDING IN REACT

```
* const Body = () => {
    const searchTxt = "KFC";
    return (
      <>
      <div>
        <input type="text"
          placement="search"
          value={searchTxt}
        />
      </>
    );
};
```

I have a variable 'searchTxt' and if I put that here

Then the value "KFC" will go inside my input box.

In one-way data binding information flows in only one direction, and is when the information is displayed but not updated. In two-way data binding information flows in both directions and is used in situations where information needs to be updated.

→ I'm not able to edit the value because it is a hardcoded value.

→ To change the value in the input box, we need to modify the variable searchTxt. But in input box, if we write something, it won't change searchTxt.

This is called one-way data binding.

Spiral

Teacher's Sign .....

Date.....

* How can we I change the value of ~~search Txt~~ searchTxt ?
Ans) :- write an onChange method.

onChange = {(e) ⇒ onChangeInput}

Create an onChangeInput function - It takes a function (which is a callback fn) which have a e event.
So, whenever input is changed, this function will be called.

→ If you need to maintain a variable that changes itself, then you need to maintain a 'React-Kinda' variable.

→ React Variable

It's like a state variable

※ [Every component in React maintains a state. So, you can put some variables on to that state]
when the state variable/object changes the ~~state~~ component re-renders.
[Every time you have to create a local variable, you use state in it]

→ In react, if I want to create a local variable like searchTxt, I will create it using useState Hook

useState Hook

→ New way of creating variables ↓

const [searchTxt] = useState();

If I have to create local variable in react, you need to use state variables.
→ State variables are created using useState hook

Teacher's Sign.....

Date.....

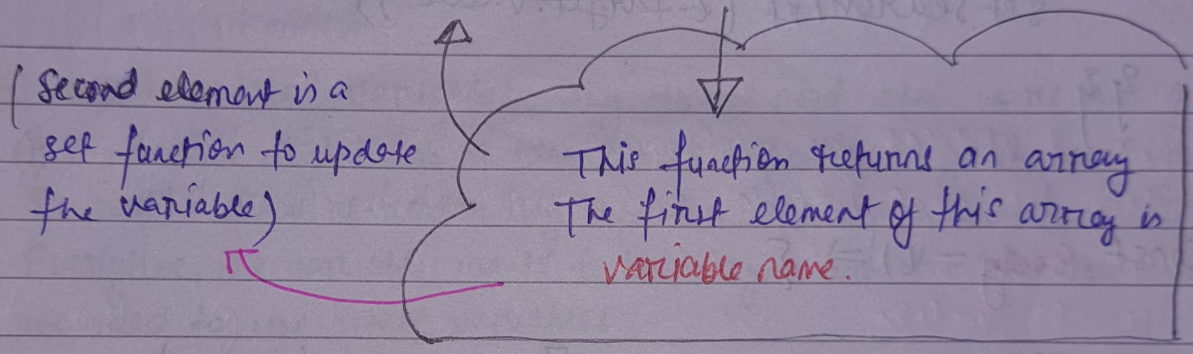
What is Hooks ?

- Hooks are normal functions.
- I get useState hook from 'react' library.
  (imported using named import)

What is the function of useState ?

- It's to create state variables...

```
const [searchTxt] = useState();
```



'searchTxt' is a local state variable.

→ To give a default value to my useState variable, do this ↓

```
const [searchTxt] = useState("KFC")
```

→ This is how we create a local state variable in react

On javascript, we create a variable searchTxt having a value "KFC" like this ↓

```
const searchTxt = "KFC";
```

Date.....

→ In react, to modify the variable 'searchTxt' I have to use the function.

useState() gives us two function.
Let us call that function 'setSearchTxt()'

```
const [searchTxt, setSearchTxt] = useState("KFC")
```

onChange = {(e) => {   ← from this event property I can read whatever I'am typing.
    setSearchTxt(e.target.value);
}}

* const Body = () => {

```
const [searchTxt, setSearchTxt] = useState("KFC");
return (
  <>
    <div className="search-container">
    <input type="text"
      className="search-input"
      placeholder="search"
      value={searchTxt}
      onChange={(e) => {
        setSearchTxt(e.target.value);
      }}
    />
```

Date.....

```
<button className="search-btn">
    Search - {searchTxt}
</button>

</>

);}
```

TWO-WAY BINDING

Here, I'm reading as well as writing searchTxt.

※ We have local variables. Why do we need state variables?
A) :- Because, react has no idea what's happening to your local variables. So, react won't re-render any updates happening on that variable. Everytime, the variable wants to be in sync with the UI. For that, we need to use state variables.
React keeps track of state variable.

※ Whenever my variable is updated, my whole 'Body' (here) component re-renders. ie, React destroy the 'Body' component and create it again. Reconciliation (Diff algorithm) is happening behind the scenes.

※ But it just re-render that updated portion. It is very quick.

[Interesting section → 01:43:00]

Date.....

Search Functionality

① we need to filter the data
→ data here is restaurant list.
→ create a function filterData();

② Update restaurantList when we filter the data.
→ I cannot update it directly,
we've to make a state variable for this

```
const [restaurants, setRestaurant] = useState(
                                    restaurantList);
```

③ Suppose I updated my restaurant with filterdata, then I must modify this local variable, restaurant with the filter data.

FilterData function

```
filterData(searchTxt, restaurant)
```

searchTxt → my input
This is the text we have to search in
restaurant ↑

I'll search my searchTxt inside restaurant and filter data.

```
const data = filterData(searchTxt, restaurants);
setRestaurant(data);
```

Date.....

Filter Algorithm (normal js)

```
function filterData ( searchTxt, restaurants ) {
    return (
        restaurants.filter ( (restaurant) =>
        restaurant.data.name.includes ( searchTxt )
    );
}

<div className = "restaurant-list">
    { restaurants.map ( (restaurant) => {
        return (
        <RestaurantCard {...restaurant.data}
            key = {restaurant.data.id} />
        );
    }) };
</div>
```

After one search, the state got updated and again when we search keywords, it won't work.

Date.....

→ when we change the state local variable the whole component will rerender. (it will be very fast)

```
const Header = () => {

    const [title, setTitle] = useState("Food villa");

    console.log("render");

    return (
        <div className="header">
            <Title />
            <h1>{title}</h1>
            <button onClick={() => setTitle("New Food App")}>Change title</button>
            <div clanName="nav-items">
                <ul>
                    <li>Home</li>
                    <li>About</li>
                </ul>
            </div>
        </div>
    );
};
```

Output :- → render

after click → render

when you click on this button two things that are happening, first of all it triggers this function and it updates title and as an when the title updated our header component is quickly refreshed and when I say refresh it means we are triggering the reconciliation process this process of checking difference b/w two trees, difference b/w the two states of virtual dom and it quickly updates this virtual dom and then it reflected on actual dom as well

If we have to implement the same thing in JS we need a lot of code and it is not performant as well but react does it very fast

Spiral

Teacher's Sign.....

Date.....

why you have to update the whole components → because you don't know how many times you have used title in the component, so in order we re-render the whole component with the updated value of state variable.

Microservices

When you are building a big app what used to happen in older days long back there used to be single big application. This application itself used to have everything i.e api's, UI etc. It is called a Monolith architecture. A monolith architecture is a traditional model of a software program, which is built as a unified unit that is self-contained and independent from other applications. A monolith architecture is a singular, large computing network with one code based that couples all of the business together. To make a change to this sort of application requires updating the entire stack by accessing the code base and building and deploying one updated version of the service-side interface. This makes updates restrictive and time-consuming, means we are not dividing software into small, well-defined modules, we use every services like, database, server or a UI of the application, in one application file. This complexity limits experimentation and makes it difficult to implement new ideas.

Microservice also known as the microservice architecture - is an architectural and organizational approach to software development where software is composed of small independent services like database, server or a UI of the application, that communicate over well-defined API's. These services are owned by small, self-contained teams.

Benefits of microservices:

- flexible scaling
- Easy deployment
- Technological freedom (koi bhi language use kar skte hain)
- Reusable code.

Date.....

Explore the world, call the api !!

→ If we write our fetch() function to call api in body . i.e

```
const Body = () => {
    const [restaurants, setRestaurants] = useState(restaurantList);
    const [searchText, setSearchText] = useState("");

    fetch()

    return (
      <>
        <div className="search-container"> ..... </div>
        <div className="restaurant-list">
          {restaurants.map((restaurant) => {
            return (
              <RestaurantCard {...restaurant.data} key={restaurant.data.id} />
            );
          })}
        </div>
      </>
    );
};
```

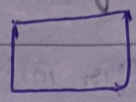
But in this implementation, everytime our UI will change (ie if we type anything in searchText) the component will re-render itself and fetch() function will be called. But we don't want our API to be called this much time.

Date.....

Two ways to load the page

1st way : ~~loads~~ → call API ⇒ render page

good way → Because of good UI experience

2nd way :  ⇒ call API ⇒ update page.

render something

we do mostly in React as we have very fast Reconciliation Engine

To do this we have a hook called { useEffect }

useEffect Hook is javascript function provided by react. The useEffect Hook allows you to eliminate side effects in your components. Some examples of side effects are : fetching ~~data~~ API data, directly updating the DOM.
useEffect accepts two arguments, a callback function and a dependency array. The second argument is optional

```
import {useEffect} from react
```

callback function, dependency array

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

If anything that we pass inside the [ ], it trigger the callback function when that this changes.

eg

```
useEffect( () => {
    console.log("searchTxt changed")
}, [searchTxt] );
```

→ It will called first time when initial render and it will be called everytime when this searchTxt changes.

Date.....

So,

empty dependency array ⇒ once after render (callback func called)] bs ek baar render hoga

dep array [searchTxt] ⇒ once after initial render + everytime after my searchTxt changes.

Two examples :-

① st example

```
const Header = () => {

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

    console.log("render");
}
```

Output →
render
useEffect.

② nd example

If we do not pass empty dependency array then the useEffect runs everytime when the UI is rendered.

```
useEffect(() => {})
```

So we will ~~else~~ call our API in useEffect hook.
So, if we use

```
const Body = () => {

    useEffect ( () => {
        //API call
        getRestaurants();
    }, [ ]);

    async function getRestaurants() {

        const data = await fetch(
          "https://www.swiggy.com/dapi/restaurants/..." );
        const json = await data.json();
        console.log (json);
    }
    ~~console~~
    return (
        < >
        ;
        </ >
    );
};
```

→ This won't work as our browser doesn't allow it due to security purpose.

→ To bypass this add CORS extension (to call api call to swiggy)

Date.....

Our state changes after two things :-

1. on changing the state local variable
2. on changing the props (that we pass during calling functional)

So in order to explore the world, we will get our Restaurant data from calling the Swiggy API.

```
const [Restaurant, SetRestaurant] = useState([]);
useEffect(() => {
    getRestaurants();
}, []);

async function getRestaurant() {
    const data = await fetch(
        "https://swiggy/api-----"
    );
    const json = await data.json();
    SetRestaurant(json?.data?.---);
}
```

→ Next page

Optional chaining

Optional chaining : Optional chaining is good way of accessing the object keys, it prevents the application from being crashed if the key that we are trying to access is not present. If the key is not present then instead of a key error, it returns undefined.

Teacher's Sign .....

Spiral

Date.....

Q. what is the use of const json = await data.json() ?

Solut The data object, returned by the await fetch(), is a generic placeholder for multiple data formats (Basically aur bhi bohot saare objects/data honge usme saath, json k saath). So we can extract the JSON object from a fetch response by using await data.json(). (hum us mein se apna JavaScript object nikaal sakte hai using await data.json()).

data.json() is a method on the data object that lets you extract a JSON object from the data on response.
The methods return a promise because we have used await keyword. so data.json returns a promise resolved to a JSON object.

So Now our API works fine but when we load our page while we fetch API from Swiggy our page render looks very bad (nums, blank ...) so here Shimmer UI comes into the picture.

Shimmer UI :- shimmer UI is a great way of loading the applications. Instead of showing a loading circle we can design a shimmer UI for our application that is good for user experience.
actually shimmer UI resembles's 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 (while UI currently loading) when a page full of content/data takes more than 3-5 seconds to load. Shimmer UI is good for user experience

Teacher's Sign .....

Spiral

Now we have designed a shimmer UI for our application Now we want to render it before our (API) called render/ page loads. To do that we need to use Conditional Rendering.

Conditional Rendering :- A conditional rendering is a way of rendering components based on a state. If the condition is true for a component, then it gets rendered; otherwise, the other component is rendered.

For example: we load a shimmer UI before our component is loaded completely. we can create a state variable that will keep the value of our current application state. i.e.

```
const [isLoaded, setLoaded] = useState(false)
```

In the above example, we are creating a state variable that is initially set to false, since our data has not been loaded in our application yet.

Until our data is loaded completely we can show a shimmer UI to the user and when our data gets loaded we can render the data on the page. The conditional rendering is done via a ternary operator [?:] for example:

```
isLoaded ? <Body /> : <Shimmer />
```

In the above expression, if isLoaded is set to false then, shimmer component will be loaded, when the data loading is completed, the body component will be rendered.

Date.....

Q. what is the difference b/w Js expression and Js statement?

Js expression :- returns a value that we use in the application.

```
1+2    // expression
"foo".toUpperCase()    // expression 'FOO'
console.log(2)    // logs '2'
isTrue ? true : false    // returns us a true or false value based on isTrue value
```

Js statement :- doesn't returns a value

```
let x;    // variable declaration
if() {}    // if condition
```

If we want to use Js expression in JSX, we have to wrap in

{/* expression slot */}.

If we want to use Js statement in JSX, we have to wrap it like

Js statement

```
{
  a=10;
  console.log(a);
}
```

⇒

Js in JSX

```
{
  ((a=10), console.log(a))
}
```

So if want to use conditional rendering using if-else in JSX.

→

```
{
  (if (isLoggedIn) {
      return <UserGreeting />;
  } else {
      return <GuestGreeting />;
  })
}
```

Spiral

Teacher's Sign .....

But now our search function stopped working so I have to make some changes in the Body components to make it work.

```
const Body = () => {

  const [isLoaded, setIsLoaded] = useState(false);
  const [searchTxt, setSearchTxt] = useState();
  const [filteredRestaurant, setFilteredRestaurant] = useState([]);
  const [AllRestaurant, setAllRestaurant] = useState([]);

  useEffect( () => {
      getRestaurants();
  }, []);
  async function getRestaurants() {
      const data = await fetch(
        "https://www.swiggy.com ....."
      );
      const json = await data.json();
      setAllRestaurant(json?.data?.cards[2]?.data?.cards);
      setFilteredRestaurant(json?.data?.cards[2]?.data?.cards);
      setIsloaded(true);
      console.log(json);
  }

  return isLoaded ?
    ( <>
        <div>
            <input
                type="text"
                className="search-input"
                placeholder="Search"
```

```
value = {searchTxt}
onChange = {(e) => {
    return setSearchTxt(e.target.value);
}}

/>
<button
    onClick = {(e) => {
        setFilteredRestaurant(getData(searchTxt, AllRestaurant
    }}
>
Search
</button>
</div>
<div className="restaurant-list">
    {filteredRestaurant.length === 0 ? (<h1>No WAYYY!</h1>) :
    (filteredRestaurant.map((rest) => {
        return <RestaurantCard {...rest.data}
            key={rest.data.id} />;
    }))}
</div>
</>
): <ShimmerUI />;
};
```