Episode - 05 Let's get Hooked

→ What even we do with react, or any other framework we do the same using normal html, css & JS.

→ But frameworks like react makes our job more easier and also optimizes things behind the scenes so that our website could load & run more faster.

→ One benifit of React :- When we switch from one page to another it does refresh or load the whole page, it just efficiently loads the new components of that new page.

→ Make Seperate folders for Seperate components.

→ create a src (Source code) folder & put all your Source files/codes these.

Js files.

→ Src
- components → create file name as Same as component name. → Put all components here.
- App.Js (Here we will assemble all our components & render it into root)

→ Read file Structure blog on official react website

component files
- Header.Js → I will use this
- Header.Jsx → Hardly matters, both are Same use ~~any~~ any one.
- Header.tsx (Typescript)

From react 17 & above if you will not write import React from 'react' and use Jsx it will work fine.

Because now your Jsx is directly getting converted into Js object → done by babel.
In previous versions it (Jsx) converts into react element & then it no Js object by babel.

★ Note: ① In your component file firstly do export.

export default comp.name;

② And then do import into app.Js

{ First, Give, then take }.

★ Note: 1.) Do not put your hard coded data into your component file. Always put it into a seperate folder.

2.) Do not put your hard coded links into your component folder. Always put it into another file and the import it wherever you want to use it.

Types of Export

default

→ You can have only one default export into your Js file.

→ You can not export more than one file using default export.

→ Syntax: export default name;

Not necessary that you will have a component only here, you can export what ever you want ex → variable.

→ Import Syntax:

import name from "file path";

named

→ You can have export more than one file using named export

→ Used when you need to export more than one file.

→ Syntax:

export const name = " ";

put export before variable type of what you are trying to export

→ Import Syntax:

import { name } from " filepath";

→ How to use import into your code.

- → if it is a Js variable then write inside { }.
- → if it is a React functional component write inside < />.

Que: can I use a default export along with named export?

→ Always put component names as

const ReactComponent] each word starts from capital letter.

(R: capital, C: capital)

→ Except components everything will start from small letter.

→ Name your constants names in upper case, the constants like logo_URL etc LOGO_URL, inside your components file.

* React HOOKS: They are normal JS inbuild utility functions, which comes with some powers.

→ They are written inside react folder which is inside our node_modules

2 very important React HOOKS

useState() → you need to import it before using it.

useEffect()

→ used to create Super-powerful react variables.

→ You have to import it from "react";

→ Import it as named import. *

State variable = It stores the state of your component.

↳ its scope is inside the component.

→ creating a normal JS variable: let element = []; default value

→ creating a state variable: const [element] = useState([])

element ← state variable name

[] ← default val

***

→ whenever a state variable updates, react re-renders the component.

→ If we use any normal variable instead of a state variable, and if any changes occurs into that variable or variable gets updated react will not re-renders that component, inside which that variable is present.

const [element, setelement] = useState([]);

element ← state variable name

setelement ← A function through which we can update our state variable (i.e, element)

↗ It's not important to name it as set"state-variable-name" we can name it anything.

```
const [elem, setElem] = useState([]);
```

we can name it anything but in industry people usually names it as set-"state_variable_name".

→ In normal variable : let elem = [];

→ updating normal variable : elem.push(2); | elem = [1,2,3];

But with state variables we can not do like this, we can not update it like a normal JS variable.

→ To update it we use "setElem" function

```
setElem([1, 2, 3, 4, 5, 6]);
```

Here we pass the updated value of that variable.

→ After getting updated react will re-render that component inside which this elem state variable is present.

→ That's why we say that a state variable stores the state of that component.

★ Just like JS we also have even-listeners into react.

```
onClick = { }
```

It takes a call back function here. That function says what to do on click.

write it inside the tag where you write className

```
func one {
    return(1);
};

func two (one){
};
```

This is call back function, because it is passed as an attribute into another function i.e, two. This function will be called back by "two" (another func)

This is higher order function because it is taking another function as an input.

```
onclick = {() => {
    return (
    );
};
```

we can even write a function inside it or we can pass function name itself

* React basically makes dom operations super fast & efficient that's why react is fast or best.

→ Reconciliation Algorithm

```
const [element, setElement] = useState([]);
```

→ Here we are just destructuring the array which we are getting from useState.

or

```
const ast arr = usestate([]);
```

→ It will basically return an array with 2 values, one state variable & a function to update it.

```
const element = arr[0];
const setElement = arr[1];
```

As soon as setElement() function will run, the diffing algo. will ~~start running~~ runing. It will start comparing new virtual dom with the previous virtual dom and then it will do re-rending of the component.

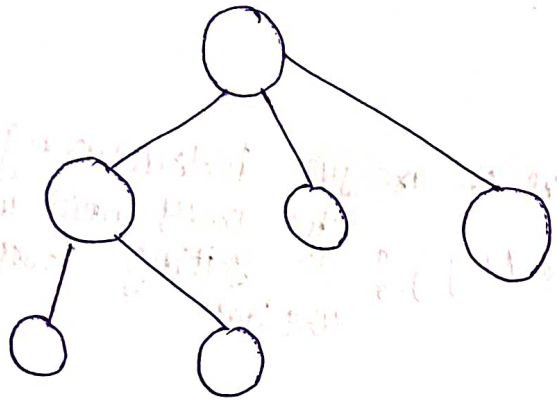
This is the reason why useState returns us an array with a state variable & a function.

* Renconciliation Algo :- It is basically a process through which react updates the browsers DOM more smoothly and more efficiently, in very less time. With this reconciliation process do Dom manipulation very fastly.

→ React uses the concept of Virtual DOM.

* Virtual DOM: Virtual DOM is like a copy or blueprint of the actual DOM.

Actual DOM



It is a tree like structure.
Every html tag is a node in DOM tree.

Virtual DOM

Virtual DOM is a representation of real/actual dom in form of a JS object.

```
{
    ____
    ____
    ____
    ____
};
```

→ React haves a copy of real dom in form of object known as virtual dom.

→ Whenever the state of the react component changes, react comes to known about it through a function which we get from useState();

→ Now after detecting the changes, react creates another new virtual dom.

→ Now react will compare the new virtual dom & the older virtual dom and it will find out the difference among them using a "diffing Algorithm."

→ After getting known the difference, (i.e, what new components or elements are added or what are deleted)

→ React will only update or render that part, not the whole component.

→ From React 16 a new technology was added into react, i.e, **React Fibre.**

→ React Fibre is a technology within the react that makes the rendering operation more smoother and efficient then ever before.

→ What React Fibre does :-

1.) Breaking down : When ever we update our component, fibre break down the changes into smaller pieces which are more manageable.

2.) Prioritizing :- Fibre has a Scheduling System through which it prioritize the changes which are needed to be render. Like essential updates will be rendered first.

3.) Rendering in parts :- Instead of rendering the whole component it will only render the essential updates in different parts/pieces as per their prority..

→ React Fibre uses a new improved reconciler.

→ The older reconciler or reconciliation process used a "stack based approach" which could struggle with a large UI

→ Now React fibre uses a fibre based data structure.

→ These fibers forms a linked list, Now as per the priority we can directly update that essential or urgent update.

→ Previously we were updating our component sequentially.

Why can't you " onClick={ setState() } " ?

React looks at whatever gets passed into onClick.

const add = (a,b) => { return a+b; }

Scenario A: onClick={add(1, 2)}

Scenario B: onClick={function() { add(1, 2) }

In scenario A we call add before onClick is ever called. Then react tries to call 3, which is the result of the add function. onClick expect a function which it can run if the button is clicked but 3 is not a function so this doesn't work.

In scenario B we declare a function in the onClick. React sees it and says "ok whenever this is clicked we'll call this function". The function is then called when the button is clicked and the function runs.

Same works with setState. You're basically calling it instead of letting the onClick call it.

Conclusion : If we call a function directly inside the onClick() (like onClick={add(1, 2)} or onClick={setState()}) then, it will first try to call that function even before the clicking on button and then, when someone will click on that button it will execute what it have received after calling that previous function, but in our case onClick={add(1, 2)} or onClick={setState()}, they both does not returns any function. So onClick will not work.

If I write add(1, 2) or setState() anywhere in my code then it will be directly executed, remember it.

If you do something like ---> onClick={setState()}

It means you are calling the setState function immediately when the component renders, not when the button is clicked. This is not the desired behavior, as the purpose of the onClick event is to specify a function that should be called when the user clicks the element.

Correct way to use onClick with setState ---> onClick={() => setState()}

The reason why onClick={setState()} is not the desired behavior is due to the fact that when React encounters this code during the component rendering, it immediately executes setState().

- onClick={setState()} // Incorrect

Above code essentially says, "When rendering this component, execute setState() and assign its result to the onClick handler, and onClick expect to get a function , so that it can execute that function when user clicks on it" It doesn't wait for a click event it immediately calls setState() during rendering.

In JavaScript, when you provide setState() without wrapping it in a function, it will be invoked immediately during the rendering phase, not when the button is clicked.

Default Export and Named Export for a same file

Yes for a single fine you can use both default and named export together

```
// Named Export
export const RestCard = (props) => {
  const { restInfo } = props;
  const { name, avgRating, cuisines, cloudinaryImageId } = restInfo?.info;
  const { deliveryTime } = restInfo?.info.sla;
  return (
    <div className="rest-card">
      <div className="rest-img-container">
        <img className="rest-img" src={CDN_LINK + cloudinaryImageId} alt="" />
      </div>
      <div className="rest-content">
        <h3>{name}</h3>
        <h5>
          {avgRating + "★"} <span>{deliveryTime + " mins"}</span>
        </h5>
        <div className="cuisines">{cuisines.join(", ")}</div>
      </div>
    </div>
  );
};
// Default Export
export default RestCard;
```

To import this we can use any one of both the syntaxes

```
import RestCard from "./RestCard"; OR import { RestCard } from "./RestCard";
```

* as export

In * as export, it is used to import the whole module as a component and access the components inside the module.

In * as export, the component is exported from MyComponent.js file like:

```
export const MyComponent = () => {}

export const MyComponent2 = () => {}

export const MyComponent3 = () => {}
```

and the component is imported from MyComponent.js file like:

```
import * as MainComponents from "./MyComponent";
```

Now we can use them in JSX as:

```
<MainComponents.MyComponent />          // As a component

<MainComponents.MyComponent2 />         // As a component

<MainComponents.MyComponent3 />
```

Note : In a same file we can have multiple react elements or components and we can use default export for one file and named export for other files. This is correct

Learn more about ES6

Learn more about difference between var and let in js