Episode - 5 Let's Get Hooked

Always keep your App.js clean.

Try to keep as less no. of lines of Code in App.js

Best Practice in industry is:-

↳ Create a Separate for each Separate Component

File structure:

- Create a Src folder, which Contains all your frontend Code.
- Create a Component folder, which contains all the Components inside your app

File naming for Component: Try to give the same for the File & Component

- Try to use 1st letter Capital while naming a Component & File

File name extensions: .js & .jsx

- Both are Same, you can use any extension
- It will give error

When You Create a Separate File, you need exports that file

export default ComponentName;

& import in main index.js / App.js file

import ComponentName from "./src/path/comp";

- import & and export name must be same.
- The name you are exporting the Same name you need to Import

mention the exact path of Component

Whenever you have hard coded data, never ever keep it in Component file.
Also, never ever use a hard coded string in your Component file

- Harded Code Keep in Separate folder eg:- utilis Folder. [common utilis]
  config.js / data.js / utilis.js etc
  constant.js / common.js
- Also it's a good habit to name your constant in Capital URL

There are 2 types of Import & Export
- Named Import / Named Export
- default Import / default Export

- **Default Export:-** export default ComponentName; [Syntax]
  - In default export you can export only one Module at a time.
    Multiple export is not possible in these case.

  *Hence Multiple Export is possible using Named Export *

- **Named Export:-**
```
export const CDN_URL = " "
export const LOGO_URL = " "
```
  Just by writing the export keyword before the Variable is named export
  With Named Export you export Multiple things from the Same file

---

- **Default Import:** import Header from "./components/Header";

- **Named Import:** used to Import only Named export.
  - Named Import import Named Export.

[Syntax]

```
import { CDN_URL } from "../utilis/constants";
import { LOGO_URL } from "../utilis/constants";
```

Use {} in Named Import to import Named exports

React Hooks:

- Are normal JS fn at the end of the day
- Normal JS utility fn

Two Vimp React Hook: (used Maximum times)
- useState()
- useEffect()

- To use useState() Hook import it first
  import {useState} from "react";
- used to create State Variable
- maintains the state of Component.

Whenever a state variable updates, react Re-renders the Component

useState Syntax:

```
const [title, setTitle] = useState(Input.Value)
```

- title: array
- setTitle: callback fn
- Input.Value: Initial Value / default value

Event Handling:

Click events.

```
onClick={() => {}}
```

Takes Callback fn in Input

There are many browser events of React such as:- onClick, onBlur, onMoveOver, onChange, etc

You can update UI using states in React.

props: props are nothing but properties.
In propes we are changing data dynamically. Very Important for Data handling.

In HTML you create html element and asaign attribute to each elements Similarly, here
In React you create custom html element (JSX) and assign props to each element

Event handling in react is done with the help of props. Props are also used for data handling.	You can update UI using states in React

- onclick = {function_Name} (Proper)
  - ↳ fn is called only when you click it, even when code is evaluted
- onclick = {function_Name()}
  - ↳ fn is always called without clicking on it, when the code is evaluted

- In JS we addEventlistener with the help of this fn, which is Imperative based approach.
- You need to define, fetch element everything
- But onClick Props is declarative approach.
  - ↳ You declare end state. [Event handling in React with the help of Props
- Event handling in React is done with the help of Props & it start with 'on' KeyWord such as → onClick, onchange, onblur etc

React will Render you DOM Single time [Only one time]
- React updates UI with the help of States → with the help of states you change the Variable Value & update on UI
  - without state, if you are trying to update the UI, by changing the variable value, it will not work.
  - In order to change your new variable value and update on UI, use states.

- If you want to use useState in code, then first import it by.→
  import {useState} from 'react';

- UseState is a React Hook
  - ↳ React Hook is a Utility fn

Syntax: -

```
const [title, setTitle] = useState(Input Value)
```

- title → variable
- setTitle → fn
- Input Value → Initialise value / Current Value
- output returns an array → updated Value of variable (1), a fn for updating the value (2)
- destructuring

```
clickHandler() {
  setTitle('AGCOE');
}
```

- States are changing per component Instance basis.

when something changes on UI
React uses Reconciliation Algorithm, also known as (React Fiber)

- Virtual DOM is nothing but a JS object (nested obj)
- Virtual DOM is not an Actual DOM
- Virtual DOM is representation of Actual DOM

- React Render cycles are very fast.
- React Renders UI very fast

Diff Algorithm: It finds out the different betn previous Virtual DOM (old Virtual DOM) & updated Virtual DOM (new Virtual DOM)

- And After finding the difference, it will Calculate the Actual different & - Update the exact Virtual DOM on Every render Cycle

- React can Efficiently findout the difference betn Virtual DOM, & update the UI

Whenever there is a change in the state variable, React will findout the diff between the Virtual DOM and it will re-render the component. It will update the DOM.

Hence, that's why React is faster → [Efficient DOM Manipulation] [becoz of Virtual DOM]

For more refer: https://github.com/acdlite/react-fiber-architecture